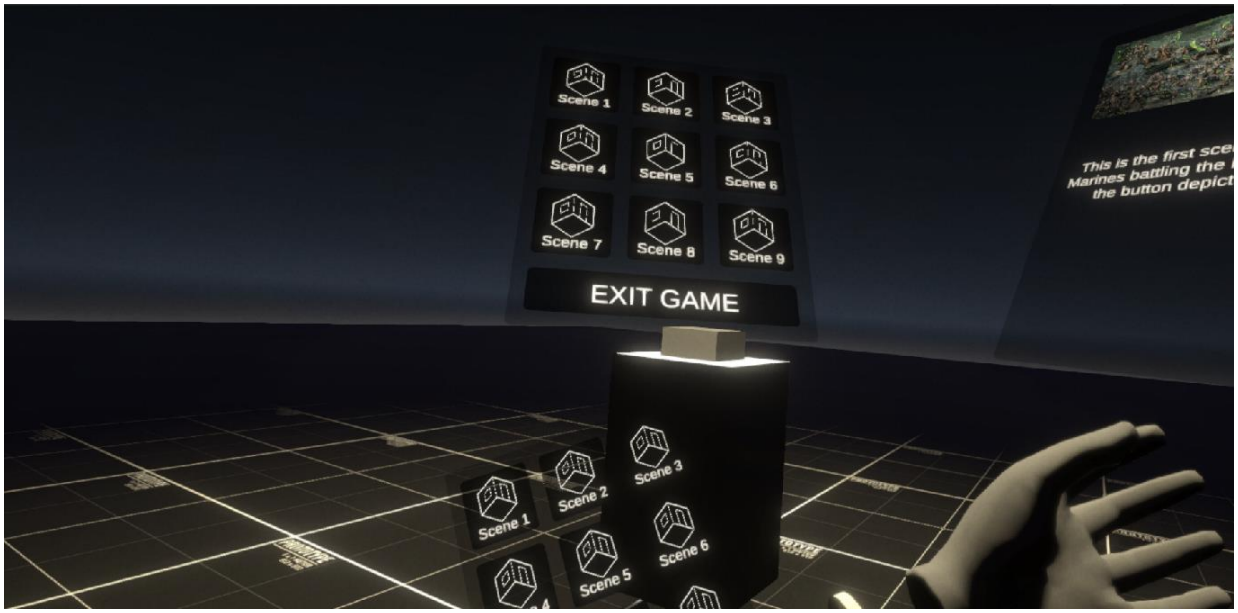
NI4OS EUROPE

OVRET SERVICE

SERVICE MANUAL OF UNITY ASSETS AND FILES



Project Overview

The following overview presents the key components of the Ovret Service project folder that we provide along with this tutorial. The main concept of how the project is structured, is that by providing a set of prefabricated Unity elements, with all functionalities integrated, a beginner Unity user, will have the ability to easily adapt and rearrange these elements to a scene to his liking, in order to adapt it to her/his research purposes. The overall methodology that is proposed to be followed is to drop required prefabs into the new scene, unpack them and adapt them accordingly.

In case of creating a new project from scratch here are the dependencies you will need to open assets from the provided project.

1.Unity Versions and packages

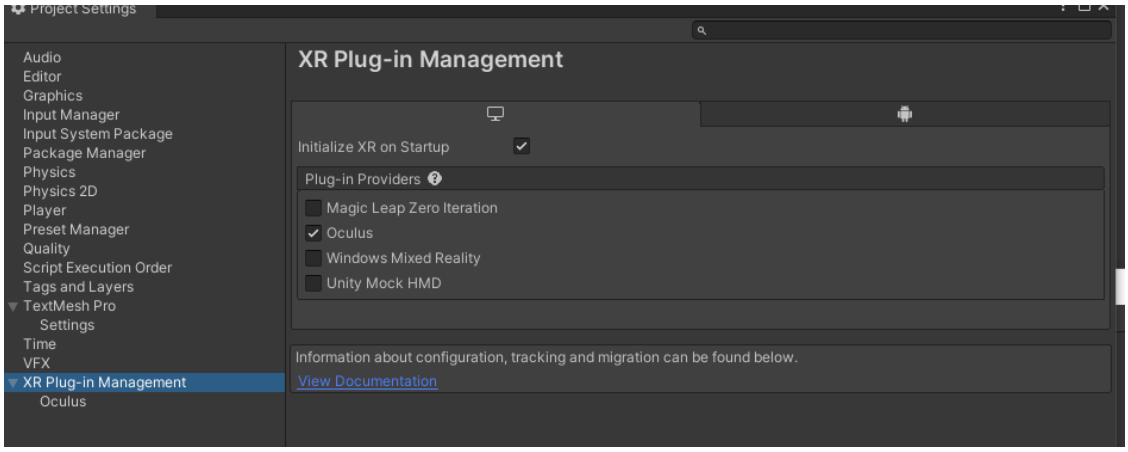
a.Unity Version

The project was created in **Unity Version 2019.4.20f1 LTS,** probably will work with any version on 2019.4 and above but might need some debugging in case of major changes.

b.Unity Player Settings

i.Case1: Oculus VR apparatus

In case of using an oculus go to player settings- project settings and go down and install the XR plug-in Management and enable your devices.



Key changes in latest version of unity!

https://www.youtube.com/watch?v=u6Rlr2021vw&list=PLrk7hDwk64-a_gf7mBBduQb3PEBYnG4fU&index=12

ii.Case2: Open XR support (beta) for Valve Index and Vive

Follow instructions from VR with Andrew

https://www.youtube.com/watch?v=DQAwRnGAHoU&t=438s

c.Package Manager packages

- XR interaction toolkit preview.7-0.10.0-preview 7
- Probuilder 4.5.0
    - https://www.youtube.com/watch?v=YtzIXCKr8Wo&t=3s
    - https://www.youtube.com/watch?v=GioRYdZbGGk
    - https://www.youtube.com/watch?v=LR83y1DZzc8&list=PL6we9_qOZgXDehm-75XZHb6ObLAzD8ilT
- ProGrids preview.6- 3.0.3- preview.6
- TextMeshPro 2.1.1 but latest should work too
    - https://www.youtube.com/watch?v=D33d4w89vTs&t=7s
    - https://www.youtube.com/user/Zolran/videos

- Universal RP 7.5.3
    - https://www.youtube.com/watch?v=HqaQJfuK_u8&list=PLbT_kxJXjbqLT8fmre3Vt--fEXnzdY2rJ
    - https://www.youtube.com/watch?v=HqaQJfuK_u8&t=1s
    - https://www.youtube.com/watch?v=WFye3k4bvEc

d.3rd party programs for material creation

- Materialize- for creating photorealistic materials in unity
    - https://www.youtube.com/watch?v=-LUlfdIZ6IM
    - https://www.youtube.com/watch?v=-LUlfdIZ6IM&list=RDCMUCgd3l8iA5zBYVa4sQ6-ONFw&start_radio=1&t=61

e.VR Development tutorials

- Part of the tutorials that explain very well how VR works and the functionalities that are integrated into the project arrive from the game Developer Valem and VR with Andrew
    - https://www.youtube.com/watch?v=NU_cLqYrYjo&list=PLrk7hDwk64-a_gf7mBBduQb3PEBYnG4fU
- Also, some elements such as URP fader and the wrist inventory system are attributed to VR with Andrew you tube channel
    - https://www.youtube.com/watch?v=yhn501096Fc&list=PLmc6GPFDyfw-LG5NUdrJcUeAU21TrrTWT
    - https://www.youtube.com/watch?v=ndwJHpxd9Mo&list=PLmc6GPFDyfw90Xo_T69Va6kw07qJ8nLz7
    - https://www.youtube.com/watch?v=OGDOC4ACfSE
    - https://www.youtube.com/watch?v=JmaAHyNvA98

f.Unity AssetStore

- **Surge** by PixelPlacement- free
- **Gridbox** Prototype Materials- free
    - Especially useful to use along with probuilder in order to quickly prototype a custom environment and scene and use Prototype style materials.

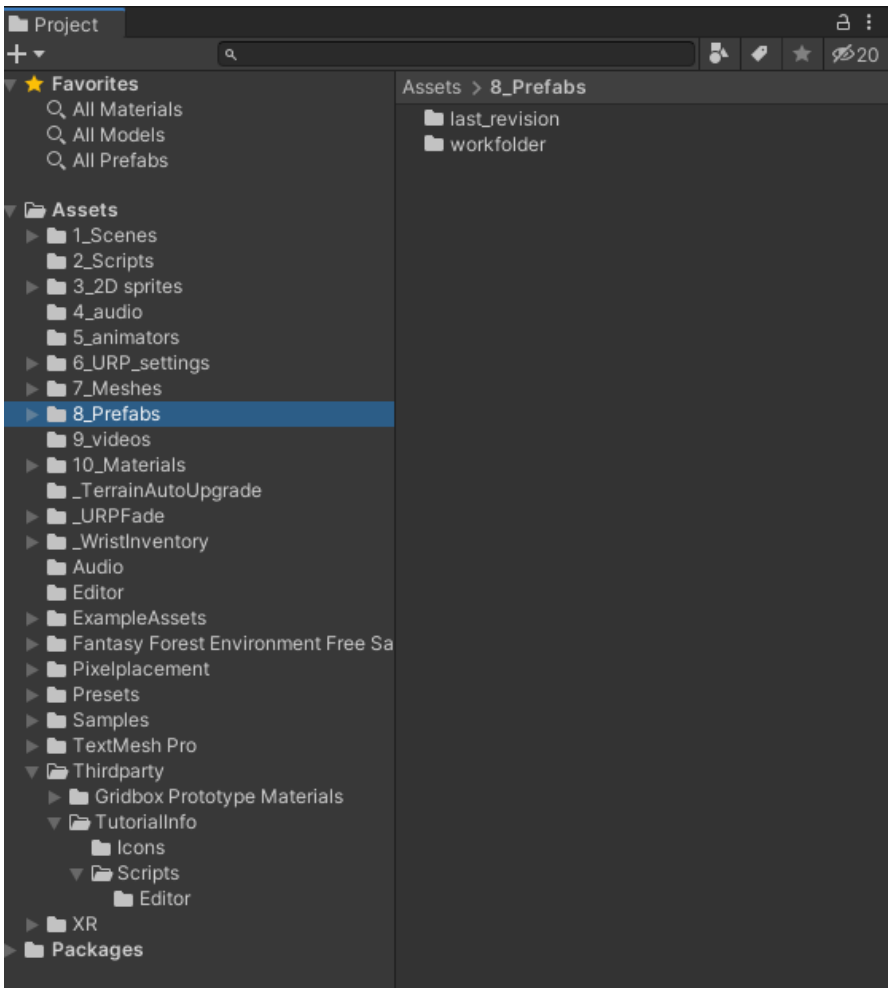
2.Scene overview

The project introduces two template scenes.

- One scene (IntroScene) to be used as a general intro Scene for someone entering the simulation in order to have an overview of all content available to visit and explore.
- A tutorial scene (LevelCreation_1) that features all functionalities that are available
- A thematic Scene (levelCreation_2) that features a thematic scene using elements downloaded from sketchfab.

The main concept of the proposed service is that we provide all the necessary prefab objects and functionalities, so that a beginner user in Unity can use these prefabs and in an easy way interact with them and edit them accordingly in order to create his own scenes, VR experiences and virtual exhibitions. Each template scene is broke down into big components with all references and functionalities integrated. By copying and unpacking the prefabs and ensuring that all references remain, a beginner user can create his own VR experience with minimum effort.

The main Hierarchy in the project folder and the Asset is as follows.

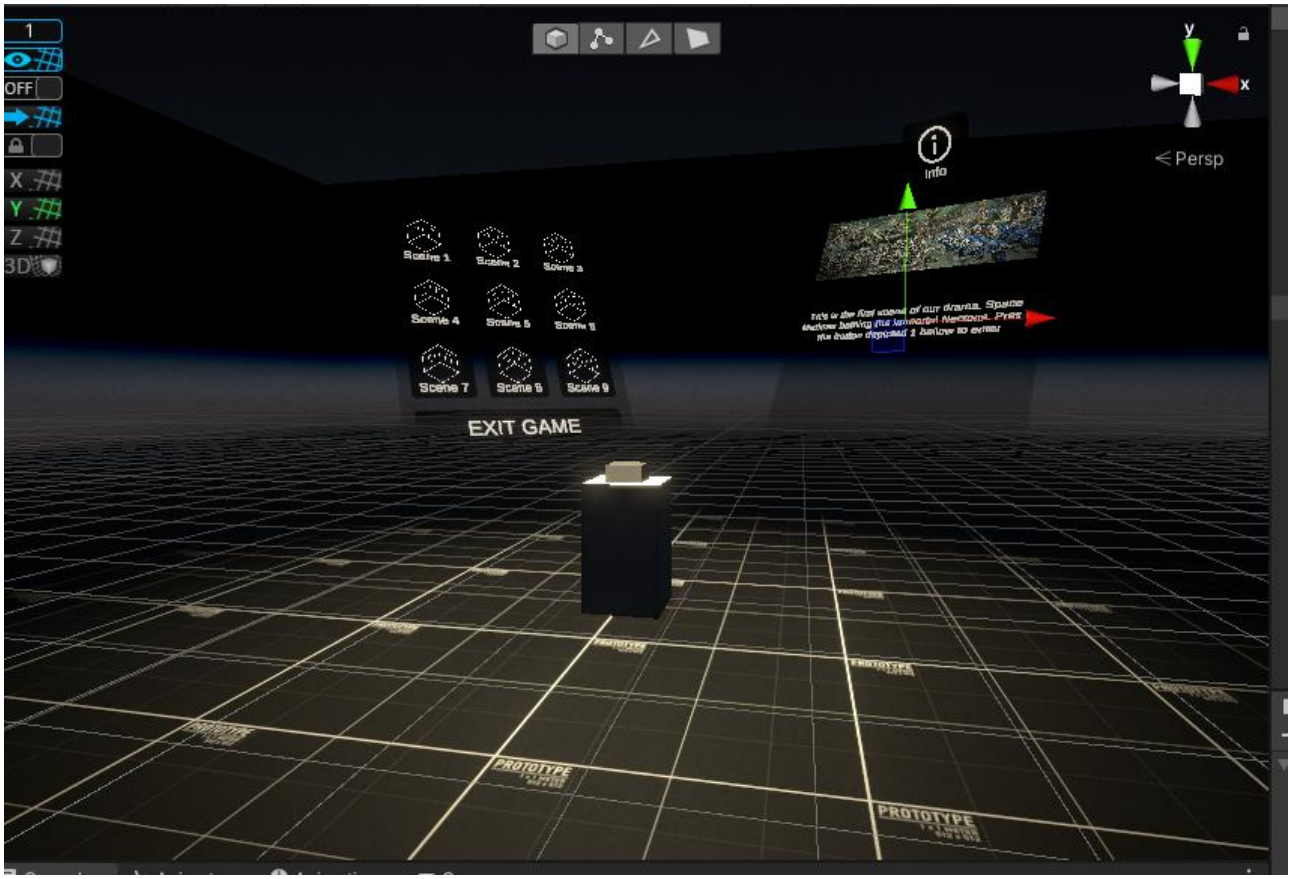


The asset folder is organized in the following order:

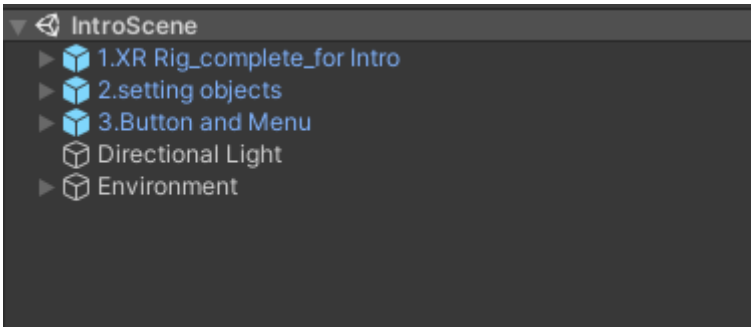
1. Scenes- all scenes in the project
2. Scripts- all scripts in the project

3. 2D Sprites- 2d sprites, icons and images
4. Audio- audio tracks
5. Animators- all animators for doors etc
6. URP-settings- key settings for use in the Universal Render Pipeline
7. Meshes- all models and meshes imported into the project
8. Prefabs- all prefabs plus previous prefabs made during the construction of the project. All features prefabs are in the last_version folder, organized per scene.
9. Videos- all videos currently on the scene
10. Materials- all materials currently on the scene.

a.Intro Scene



i.Intro Scene Hierarchy:



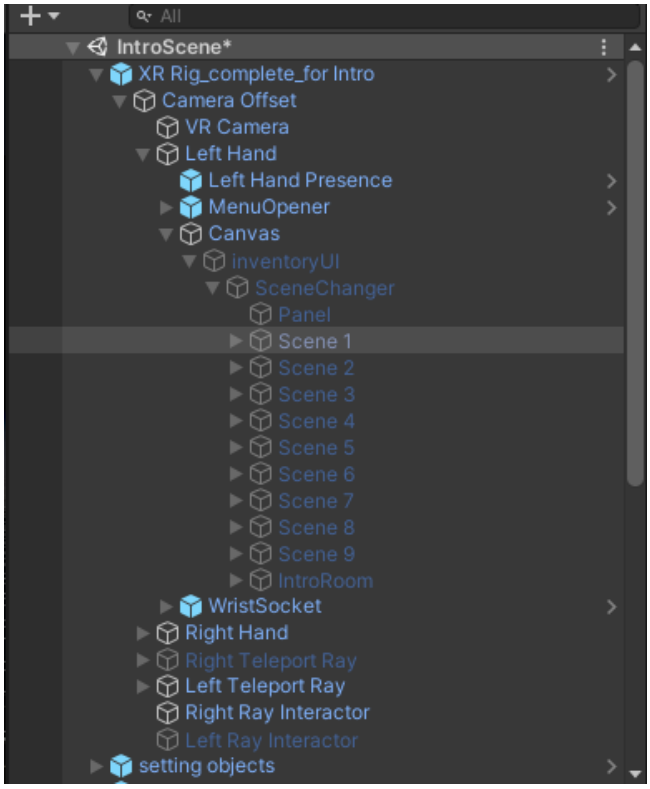
ii.IntroScenePrefabs:

The introScene consists of the following Prefabs:

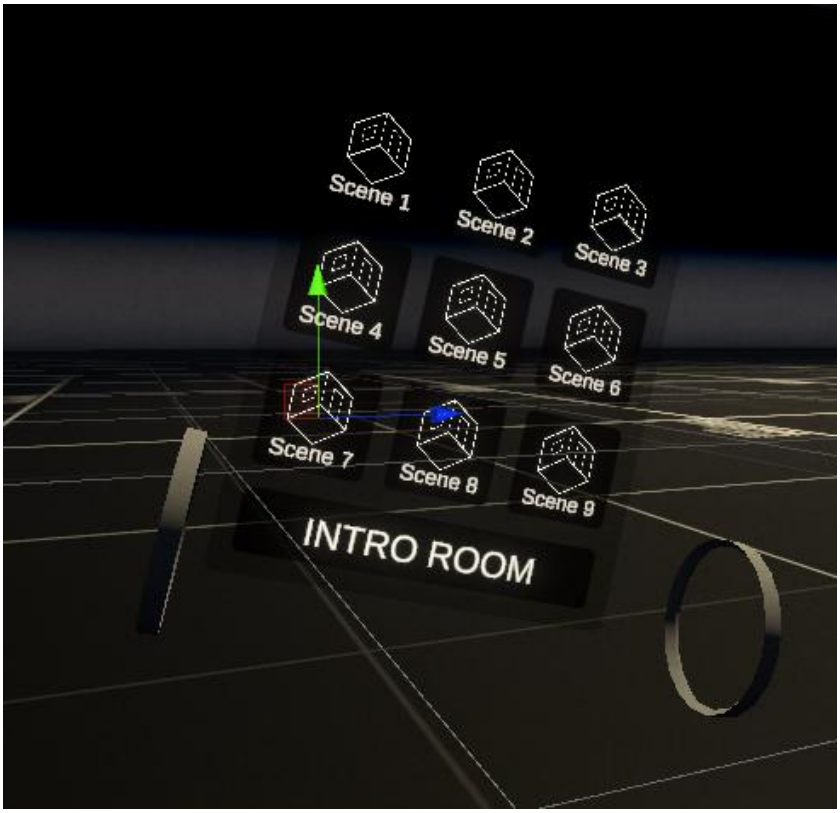
1.XR Rig_complete_for Intro

It is the **VR rig** specifically designed to be used inside the intro Scene. Key prefabs that need to be tampered with or drop them on the scene are the following:

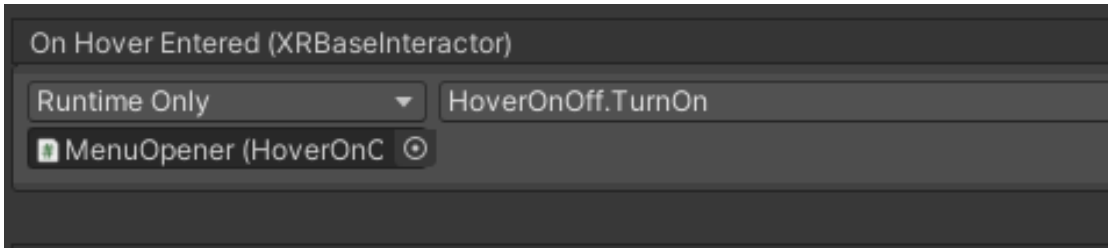
1a.Inventory UI

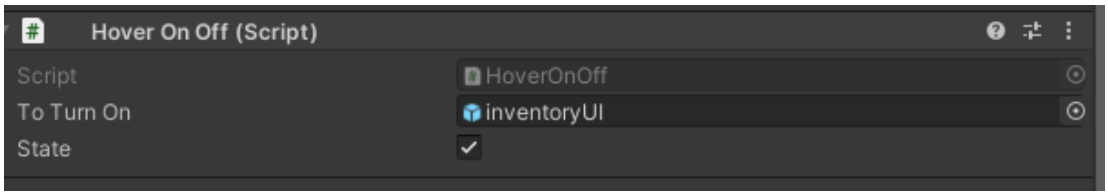


Inventory UI, is the menu that opens and closes **on hovering on the left ring far of you hand**. The hovering movement opens up a menu that controls specific functionalities incorporated into a User interface. In the case of this project we have included in the intro Scene only teleportation functionalities to jump directly to specific scenes.



The inventoryUI, is controlled by the **MenuOpener** Gameobject, that is present under the lefthand Parent gameobject. The **MenuOpener** includes an XR Simple Interactable and a **OnHover Entered** event that controls the **hoverOnOff.** The hover on off can be used as a way to turn on or off a variety of other elements in the scene per user requirements.





Each button named **Scene**, under the SceneChanger should include the script **Scene Changer.** As **an OnClick** event should be included, the method called **SceneChanger. ChangerMenuScene.** On the string input window, the user needs to manually type the name of the scene to be teleported. In the case of the featured project scene's 1 name is **"levelCreation_1"**. All scenes need to be created and stored in the Asset folder under the folder **"1_Scenes".**

Important note!

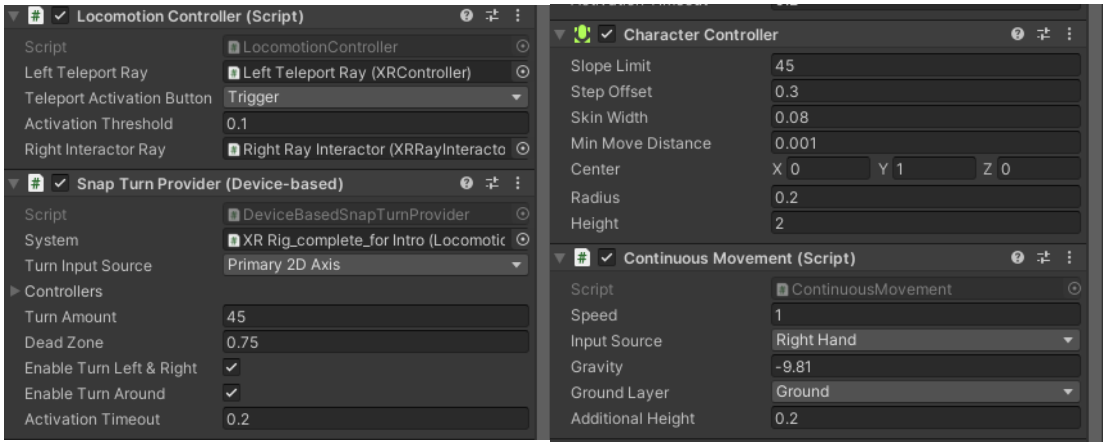
Do this for all buttons in the user interface. In case that you don't want to use the wrist, UI and use only the integrated UI inside the scene, you can just turn of the GameObject MenuOpener, with the integrated Script HoverOnOff

1b.Left Teleport Ray and Right Ray Interactor

These two components control the ray interactor to select UI elements and the way you can move around in the environment by teleportation. By default, left hand is for teleport and right hand Is for ray interaction.

1c.XR rig_complete_for Intro- snap turn and continuous movement

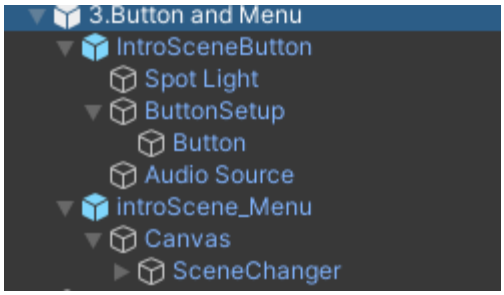
In the parent gameobject there are two scripts for controlling **snap turn per 45 degrees'** angle, that activates with the **2d Axis on the left controller** and **the continuous Movement Script** that enables to continuously move in any direction by **the 2d axis right hand controller**. In case of any feeling of discomfort these two should be the first to turn off in the inspector.



2.Setting Objects

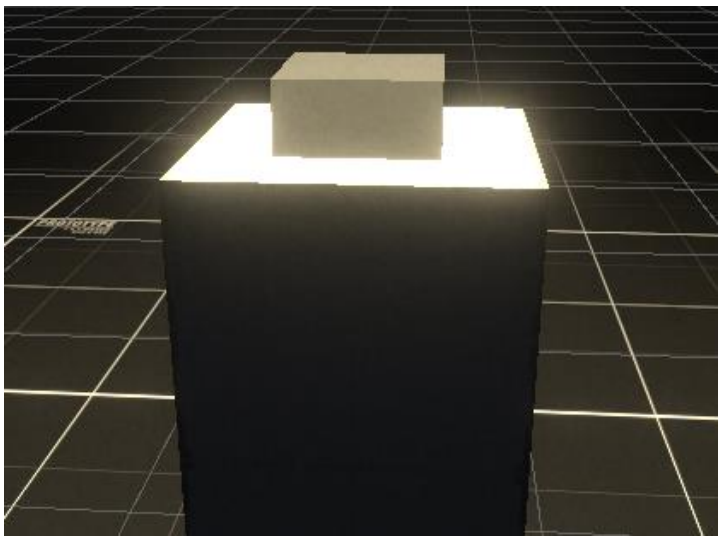
The setting object prefab is a set of objects that are needed for the scene to work properly. Specifically, **the Screen Fade**, is used for the fading effect when teleporting, **the collider Measurer** is used when using the wrist sockets as inventory system, **the global volume** uses postprocessing effects that could be further twicked to the users liking, **XR interaction manager**, is a preset object used by the system to detect XR interactions, **the EventSystem**, is a preset object used by the system to detect UI interactions.

3.Button and Menu

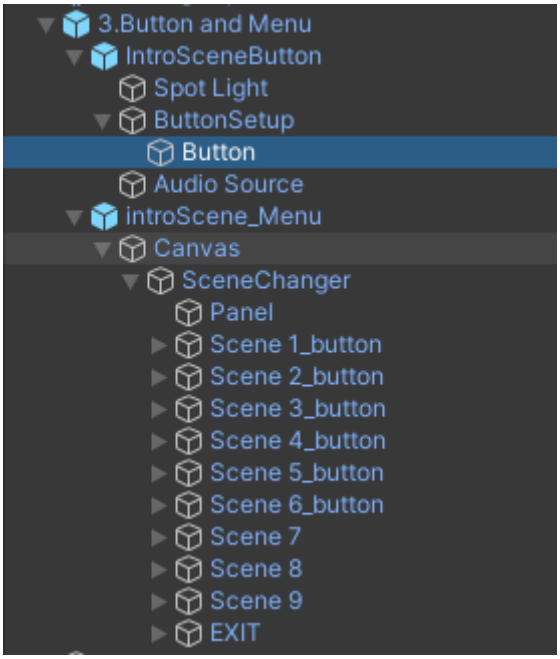


3a.IntroSceneButton

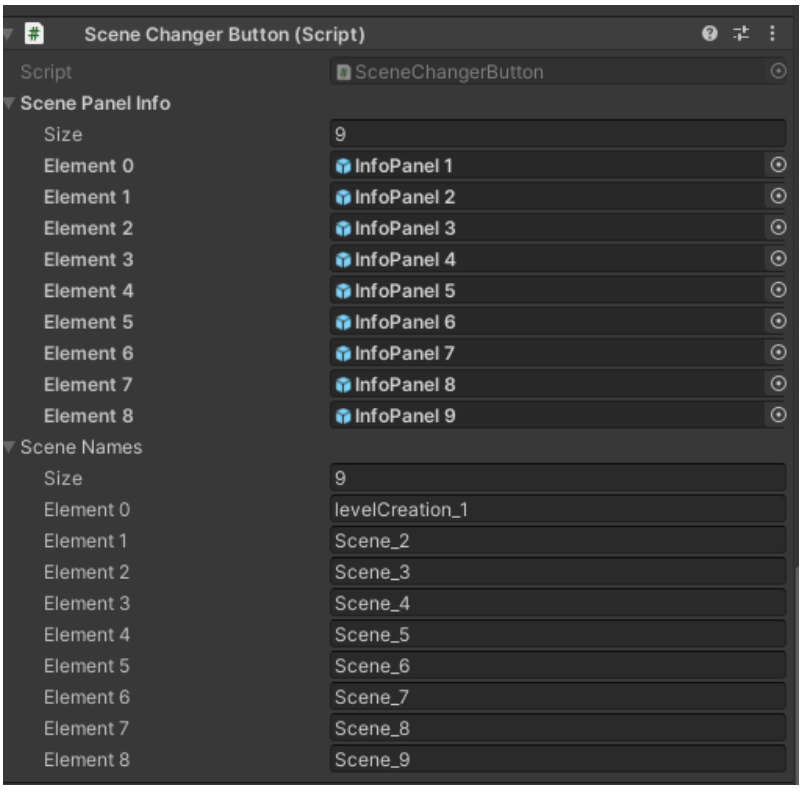
The IntroScene Button is a simple button with the only functionality attached to it, to manually enable the teleportation of the player to the selected scene after he has selected it by the overhead Panel with the Scene Selector (not the integrated into the wrist). More specifically it includes:



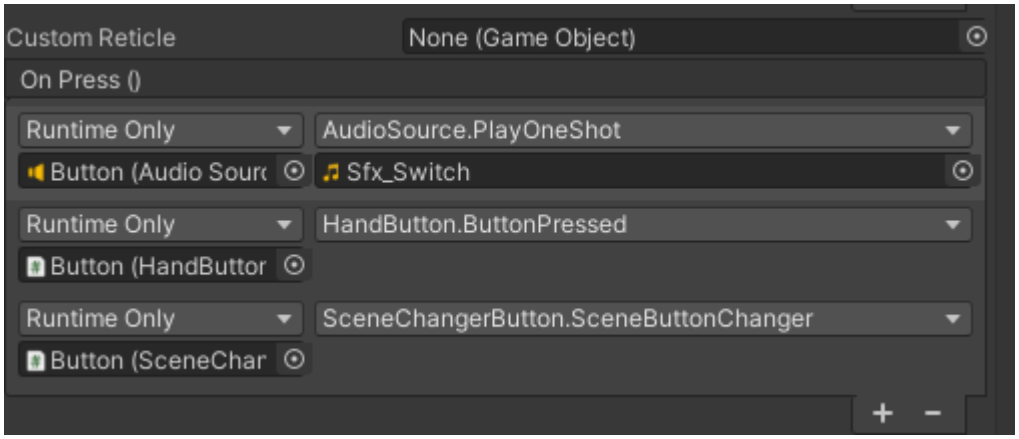
3ai.ButtonSetup Parent and Button Child component



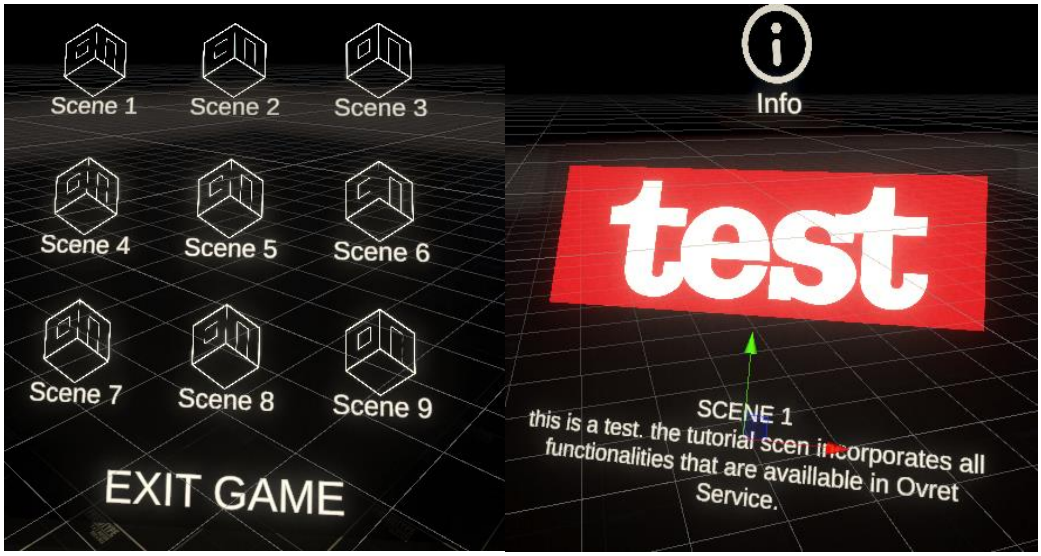
In the **button** component there is the **SceneChanger** Button Script where the user needs to manually load the **InfoPanel** for each Scene and the name of each Scene. The script checks if the info panel is on for the selected scene and loads the above scene.

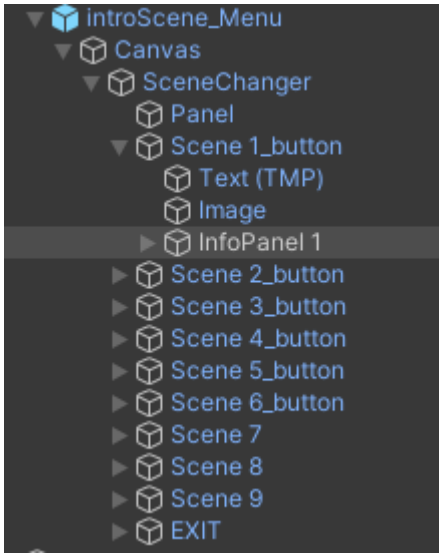


Also, on the **OnPress event manager** the button itself should be referenced and the **SceneChangerButton. SceneButtonChanger** method should be activated.



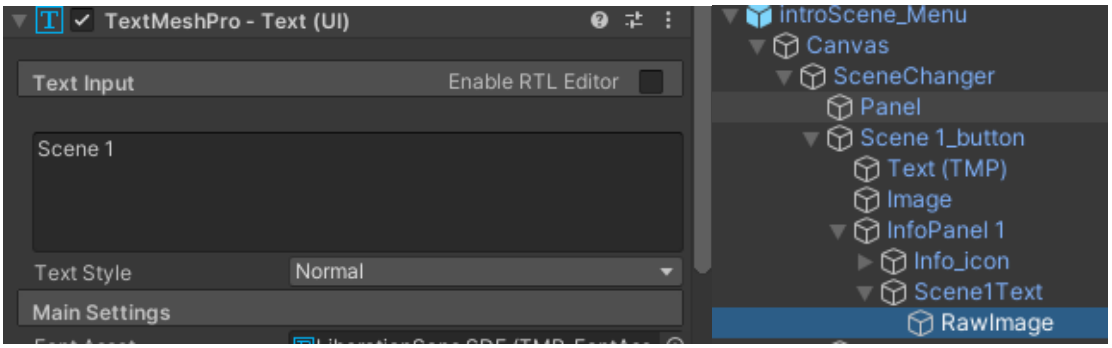
3b.IntroScene_Menu



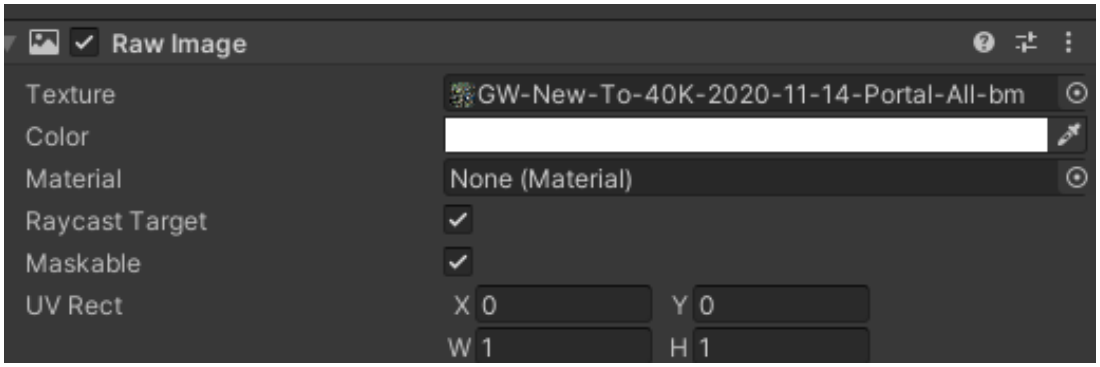


The **intro scene menu under the sceneChanger gameobhect**, includes all buttons that relate to a scene. Each button in the hierarchy includes also an **infoPanel** that starts in off_state. Child to the **infoPanel** there is a **SceneText** Object where you manually input some

information relating to the Scene Description and finally there is **a raw image** object that relates to an image that should act as a preview of the scene.



To change the **Image**, drag and drop your preferred image into the inspector in the Texture position

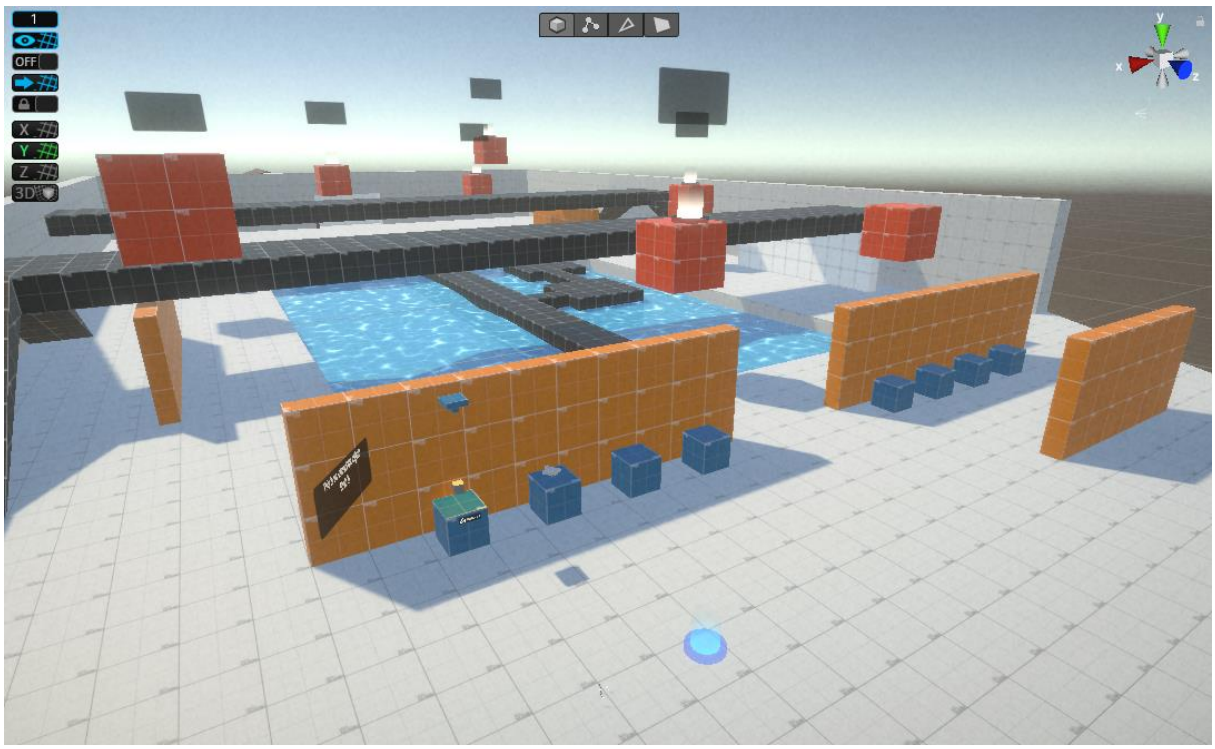


iii.Environment

The environment contains two planes the ground and the peripheral walls.

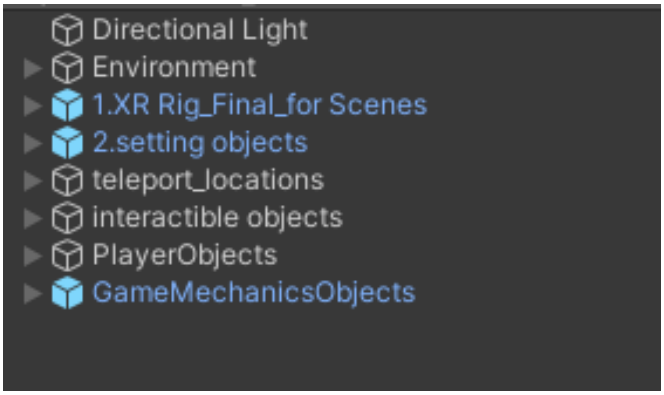
All objects in the scene are built with the built-in prototyping tool provided by unity via the asset store Probuilder and Progrids. (Check dependencies session).

b.Scene: LevelCreation_1



i.Scene Hierarchy

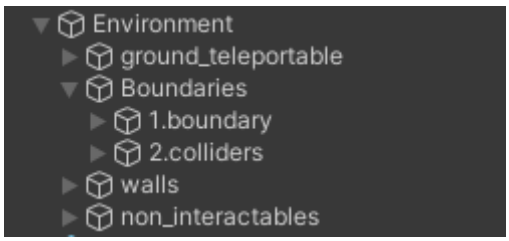
The Main Preview Game Scene consists of the following objects



ii.Scene Main Prefabs:

The Scene consists of the following prefabs:

1.Environment GameObject



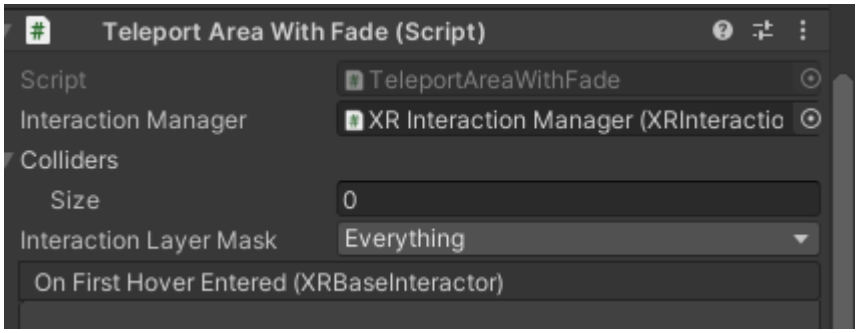
The **environment Gameobject** is a collection of all elements that consist the environment. All assets have been created with **probuilder**. In the case of assets created by photogrammetry or other modeling software such as **@Blender, @Maya, @Rhino**, it is essential to distinguish the imported models into elements/ meshes that will have separate functionalities.

More specifically:

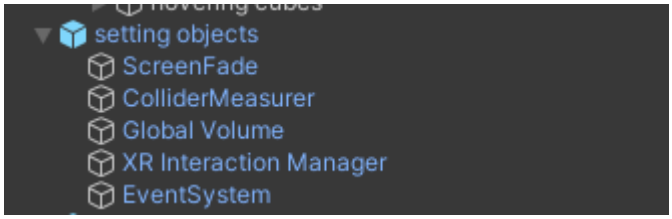
- grabable objects,
- teleportable ground and non
- interactible or teleportable objects.

1a.Ground_teleportable child object

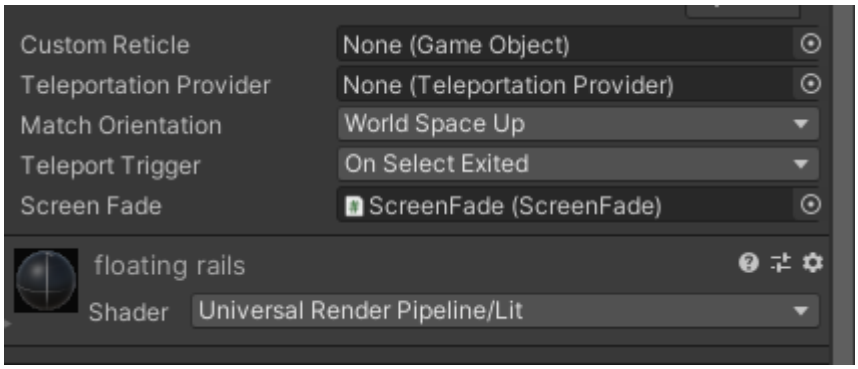
In this section there should be all objects and ground mesh that will be available for the player to be able to teleport. More specifically, each mesh gameObject that should be teleportable must have as a component the **Teleport Area with Fade Script** and **a Mesh Collider**



Check that in the interaction Manager there is a reference by the XR Interaction Manager that is part of the **settings** GameObject



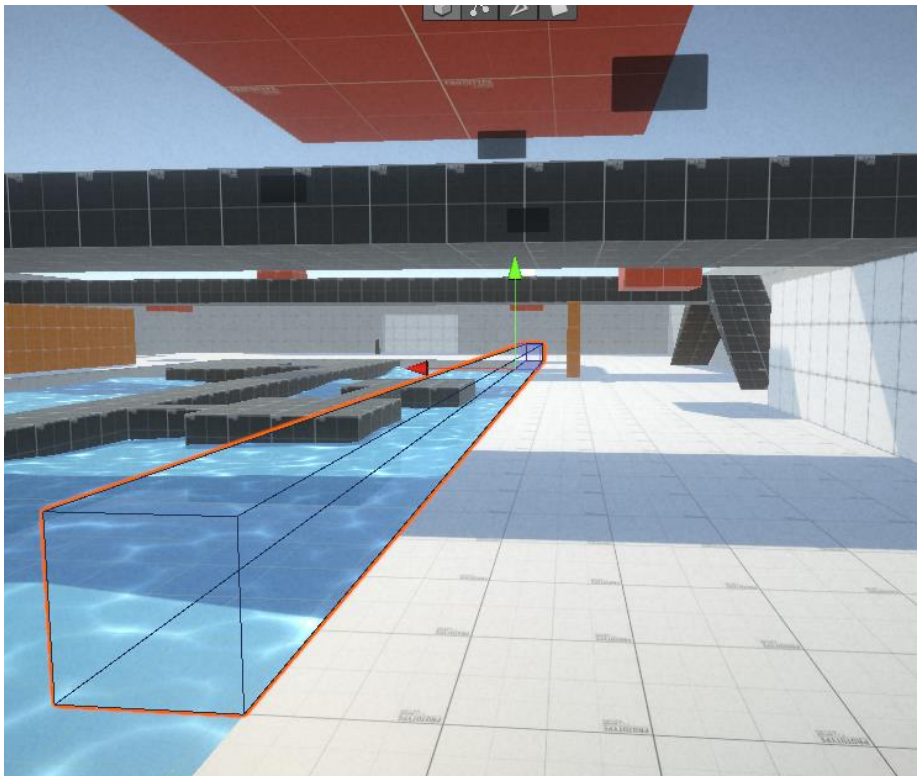
Finally, in the Teleport Area with Fade Script there should be the following references to the **ScreenFade script** part of the URP fader component.



1b.Boundaries GameObjects

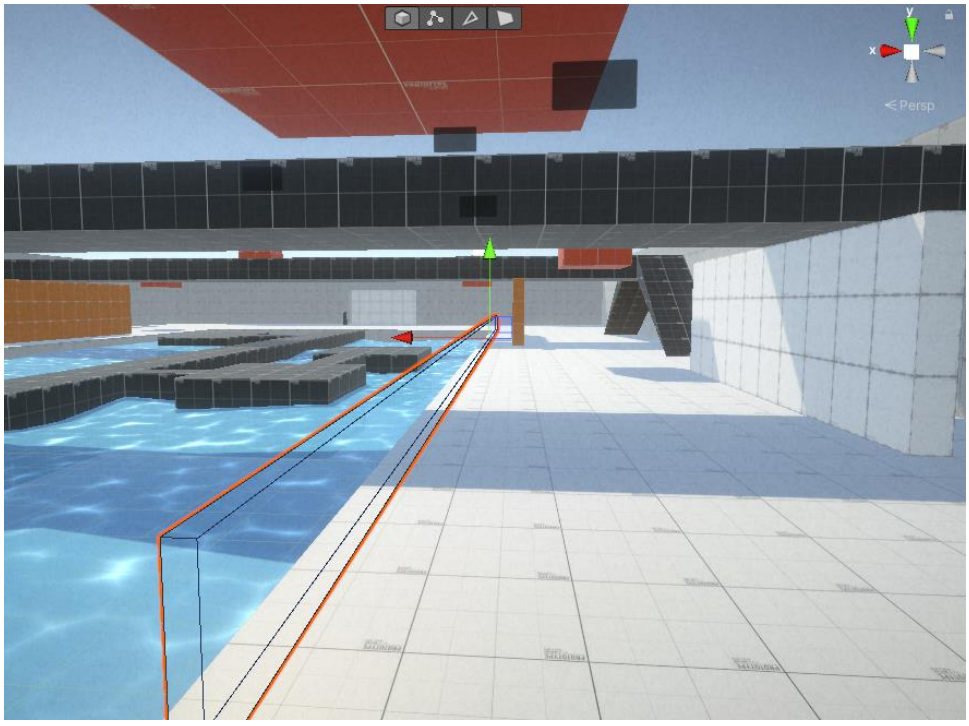
The boundaries gameObjects highlight when the Player interacts and reaches the boundaries of the scene. There are two main components. A boundary object and a collider object

1bi.Boundary object

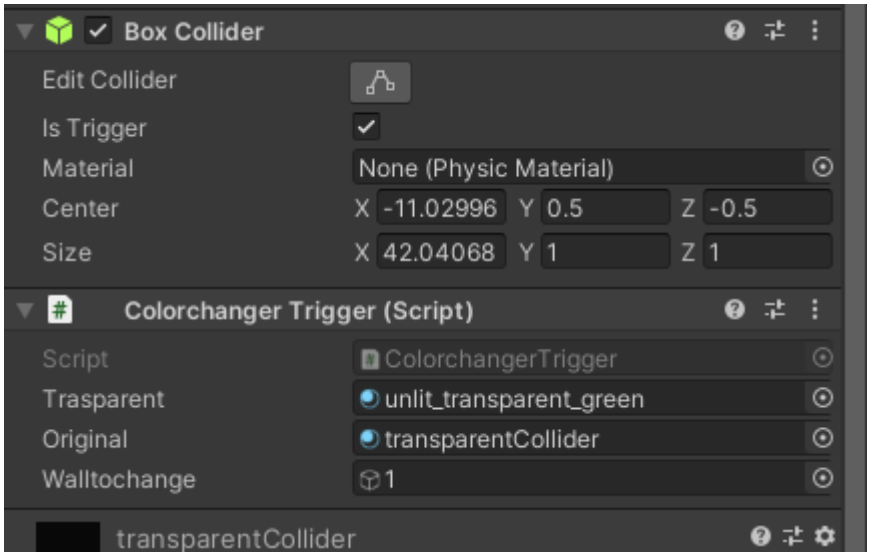


The boundary object is a simple geometry with a transparent Material and a Mesh collder and acts as a boundary for the player.

1bii.Collider Object



The **collider** object is a standard gameobject with a collider that the **Is trigger is enabled**. Finally, onto each object separately the **Colorchanger Trigger Script** is attached. The script has 3 references: 2 separate materials, one Original and one transparentCollider when the collider is triggered and the boundary object that the collider is referenced to so that when triggered it will change the material of the referenced object



2.Setting Objects

The setting object prefab is a set of objects that are needed for the scene to work properly, specifically:

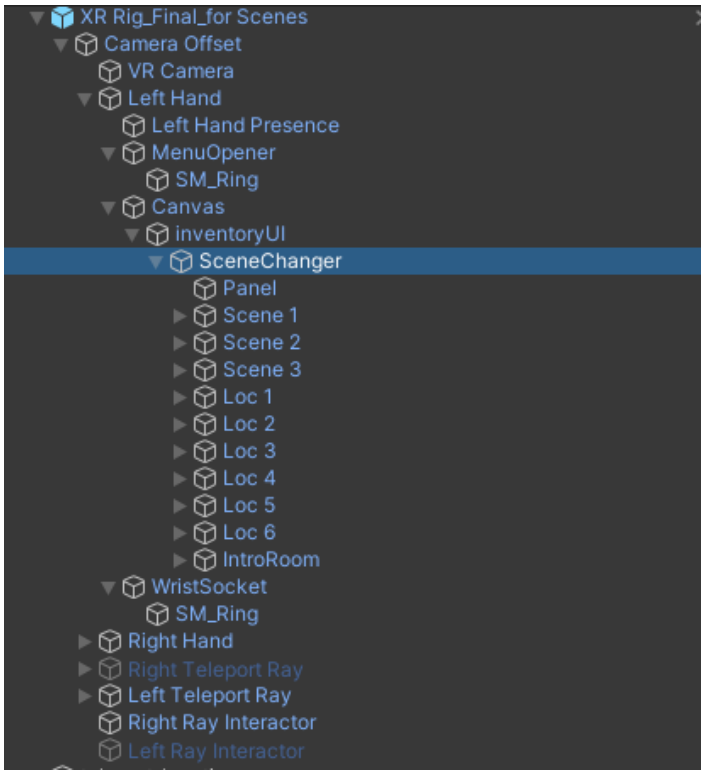
- the **ScreenFade**, is used for the fading effect when teleporting[1],
- the **collider Measurer** is used when using the wrist sockets as inventory system[2],
- the **global volume** uses postprocessing effects that could be further twicked to the users liking[3],
- **XR interaction manager**, is a preset object used by the system to detect XR interactions,
- the **EventSystem**, is a preset object used by the system to detect UI interactions.

[1] Check videos concerning URP fader on VR with Andrew
https://www.youtube.com/watch?v=OGDOC4ACfSE&t=1s
[2] Check videos concerning wrist pockets on VR with Andrew
https://www.youtube.com/watch?v=JmaAHyNvA98
[3] Check URP postProcessing effects settings and setup from Unity Guru
https://www.youtube.com/watch?v=HqaQJfuK_u8&t=24s

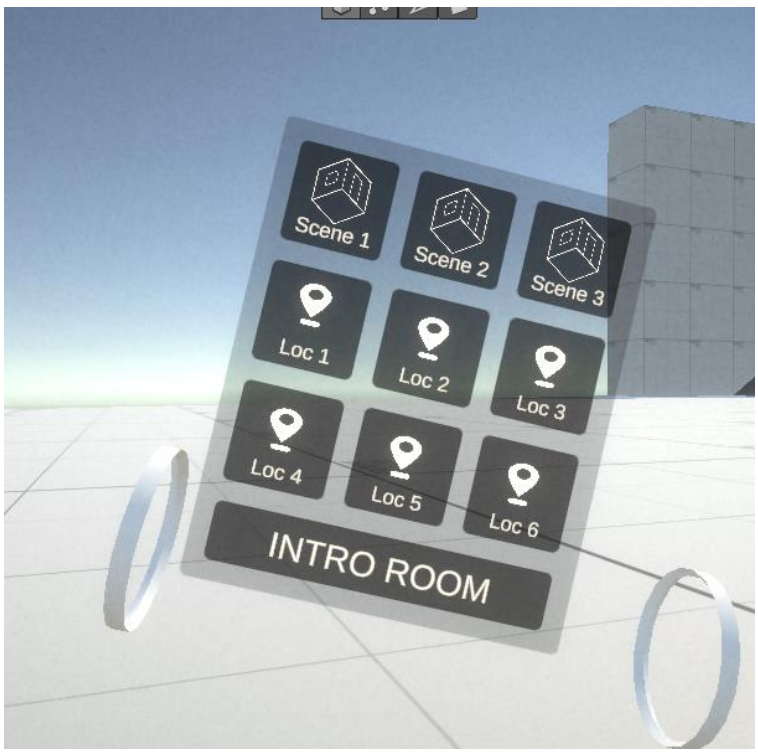
3.XR Rig_final_for Scenes

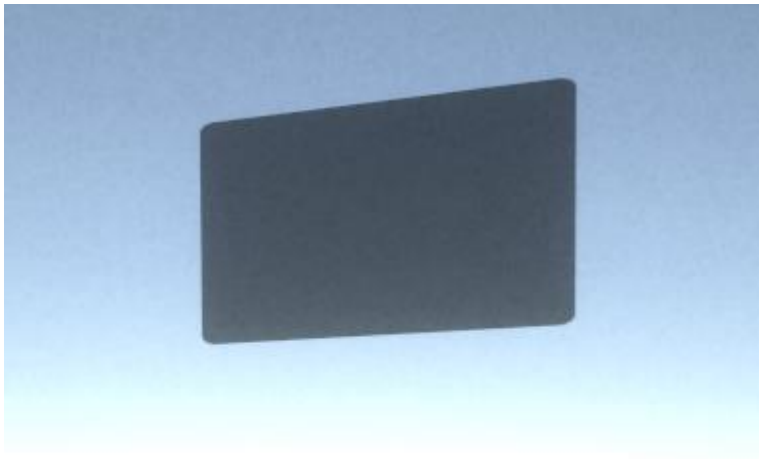
This is the VR rig same as in the Intro Scene with some small variations. The most important is listed below. Also make sure that the XR rig is set to **layer Body** and **tag Player**

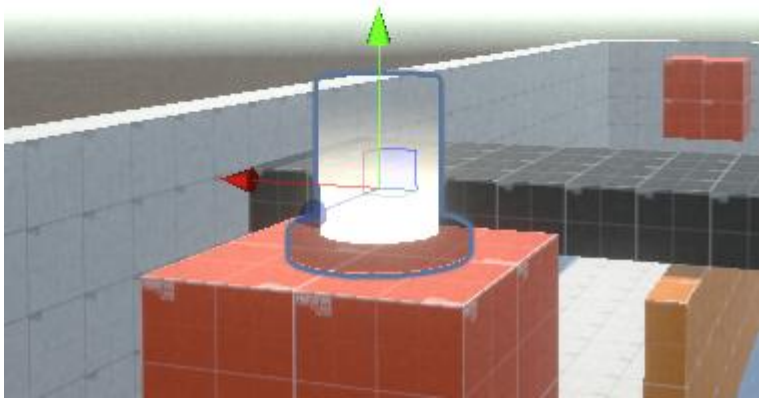
3a.SceneChanger Child Object



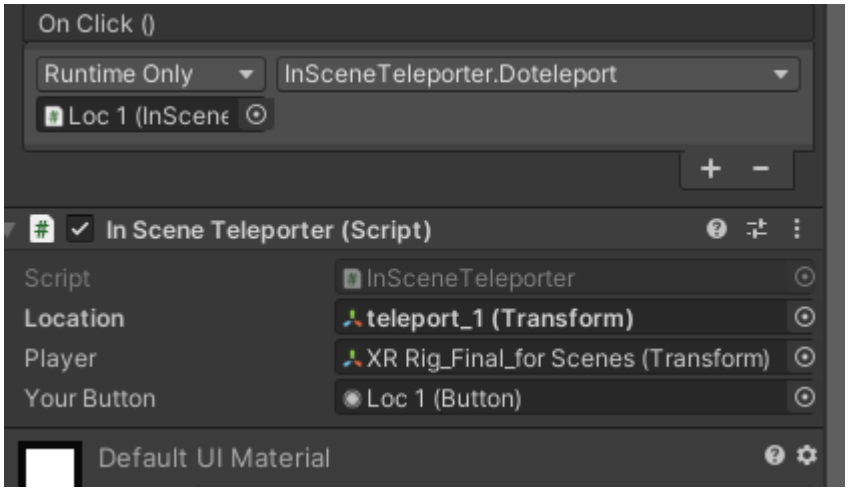
The **SceneChanger** is a variation with the scene Changer from the Intro Scene, but with one more functionality, it allows for the teleportation of the player to specific locations as seen in the word, highlighted by the rotating displays





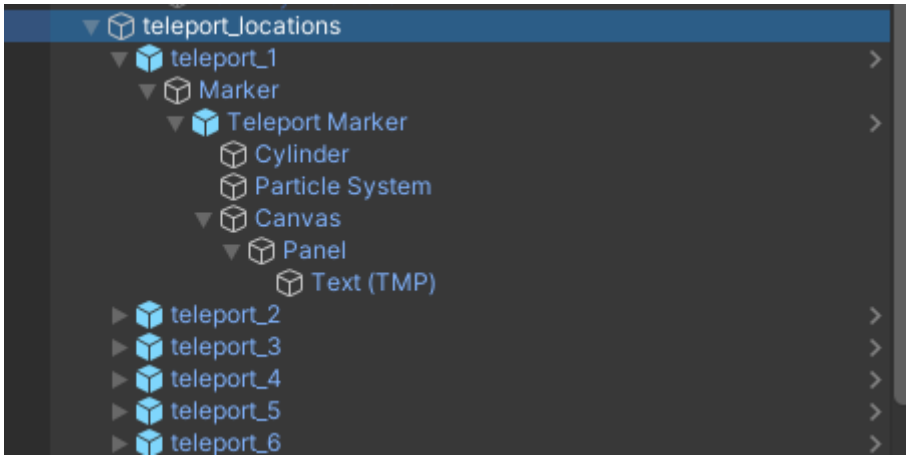


Each panel button on the User interface named accordingly Loc 1, Loc 2 etc, has an attached script the **in Scene teleporter** as seen below



In more detail, on **the on Click Event** we reference the **user interface button** so that it sees the script attached and on the script itself we reference **the transform location** of the marker (drag and drop the marker object from the hierarchy into the panel)

The teleportation Markers are below the teleport locations parent object in the hierarchy

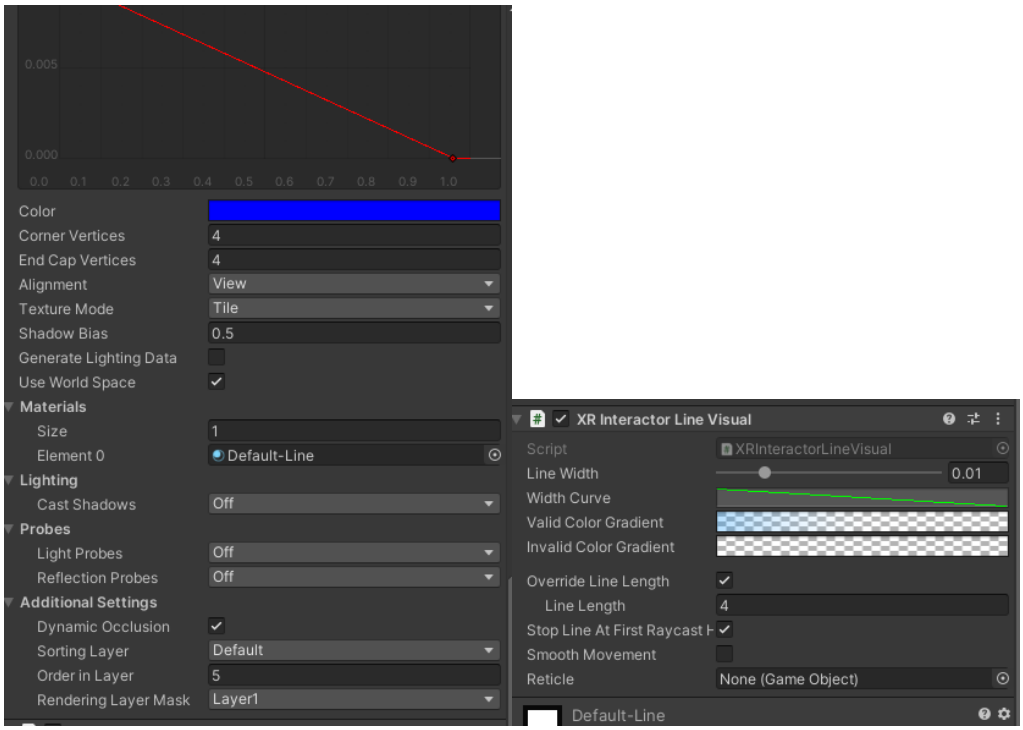


On the player reference we need to have a reference to the XR rig gameobject and on Your Button we reference the button that we will be using for the teleportation.

Each teleport marker has a script called **Todisable** attached to the Marker GameObject, that will disable the marker if the collider that is also attached to the GameObject is triggered by the Player (XR Rig, with tag "**Player**")

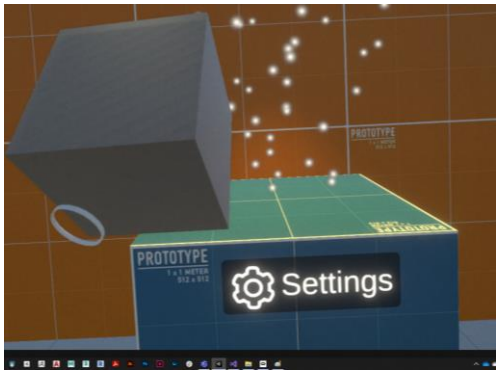
3b.Right Ray Interactor

The right ray interactor controls the ray that interacts with UI elements in the environment. To control color, width and length of the ray you select it from the hierarchy under the XR Rig_final_for Scenes and go to Ray Interactor line Visual to tweek the settings



3c.WristSocket child object.

The WristSocket GameObject works as a wrist inventory system for scaling down any object thrown inside your wrists.



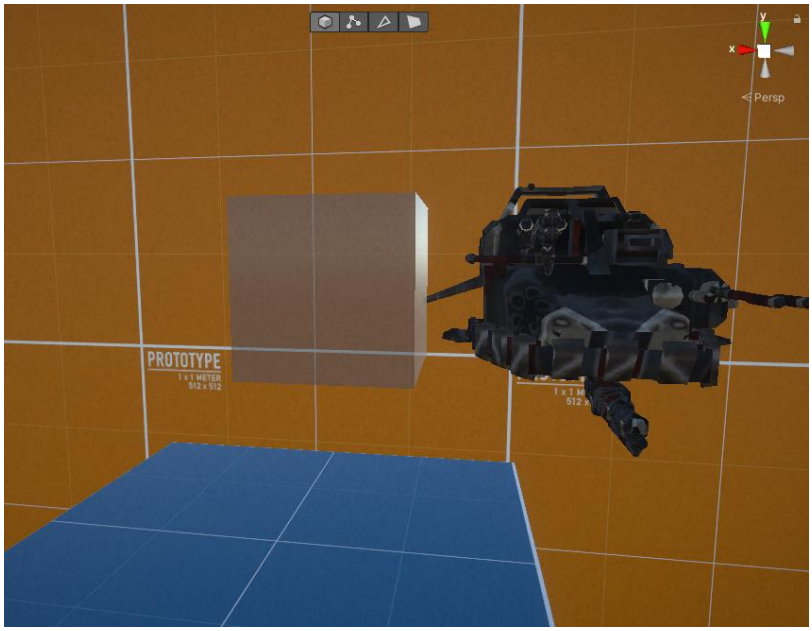
After they are inserted inside it can be grabbed again and manipulated by the player.

!!!Important notes about the WristSocket-single vs multi mesh object functionality!!!

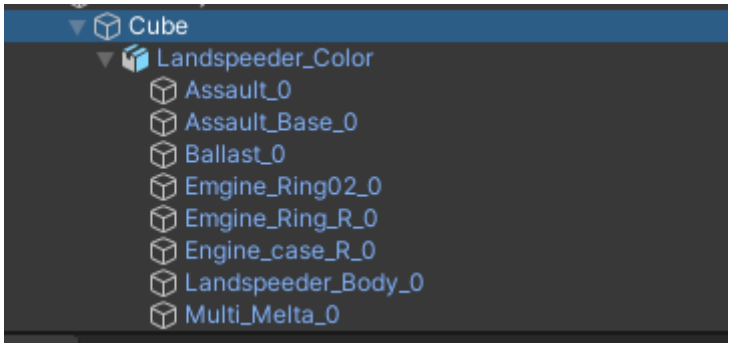
The script and the mechanics only work properly for single mesh objects. That means that the object must be a single mesh as geometry and not a compilation of different objects that are joined together, while retaining their topological independence.

In the case of as multi mesh object the developer needs to create a manual workaround.

- Step1- create a cube or a sphere or a cylinder and scale it appropriately in order to be as close as possible to the size of the object we want to use as an interactible object.



- Step2 child the object that you want to use onto the cube or cylinder by dragging in into the hierarchy

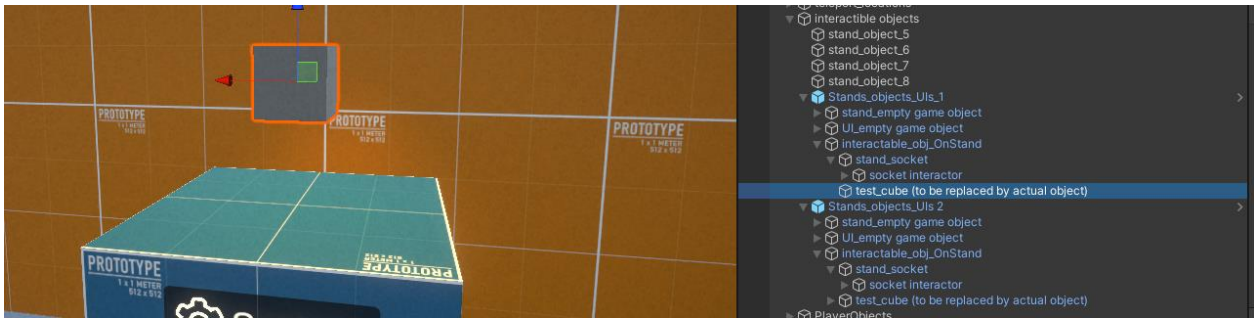


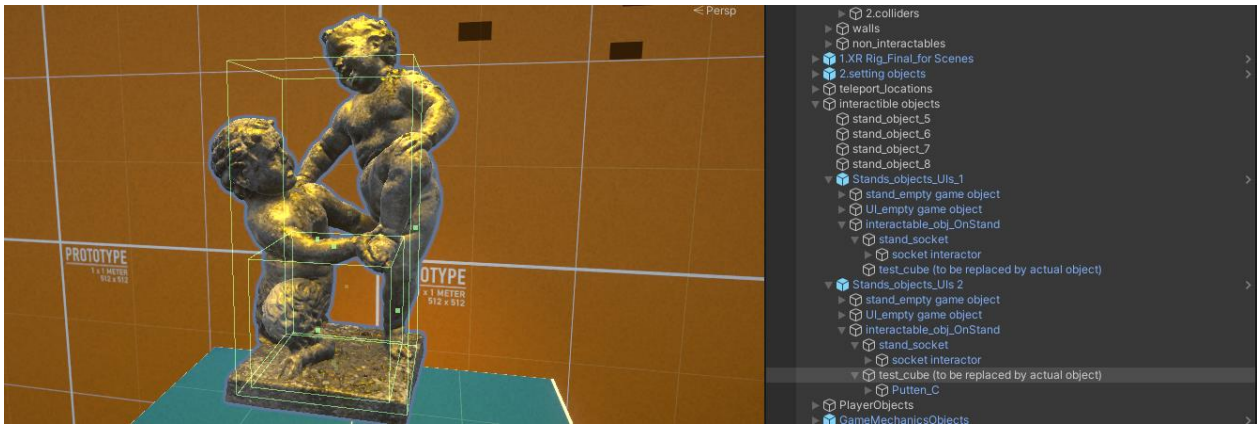
- Step3 turn off the Mesh renderer component on the parent gameobject
- Step4 add on the parent cube component the xrGrabInteractible
- Step5 set the parent gameObject layer to Grab.

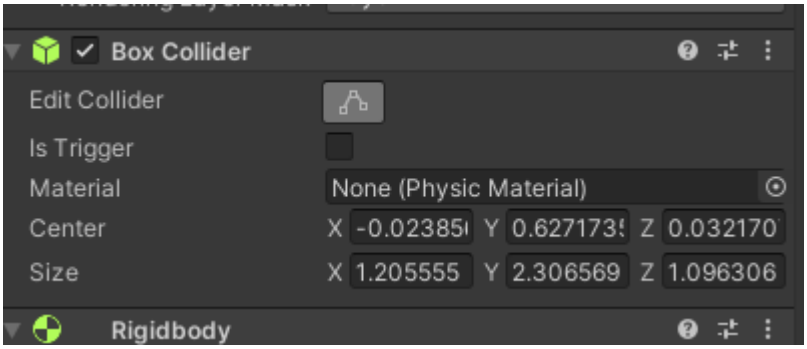
By this process, the mechanics will detect the parent cube mesh gameobject and scale according to this and as the object that we want to interact with is a child to the parent cube game object will inherit all scaling calculations done to it.

Note for users!

The scene as it is now organized there is a specific spot in the hierarchy where a Test Cube is inserted as an interactable object. In order to Facilitate the user, just add as a child to it your new model, and just edit the collider. For an overview of the Interactible object functionalities and elements check following sections **4. Appendix.Setting up an object to be used into the scene and be grabbable by the player** and Section **5. Interactible objects.**







!!!Important notes!!!

In case of an **offset grab point** there might be problems with the scaled down object appearing of centered.

4. Appendix. Setting up an object to be used into the scene and be grabbable by the player

Step 1. Make object interactable

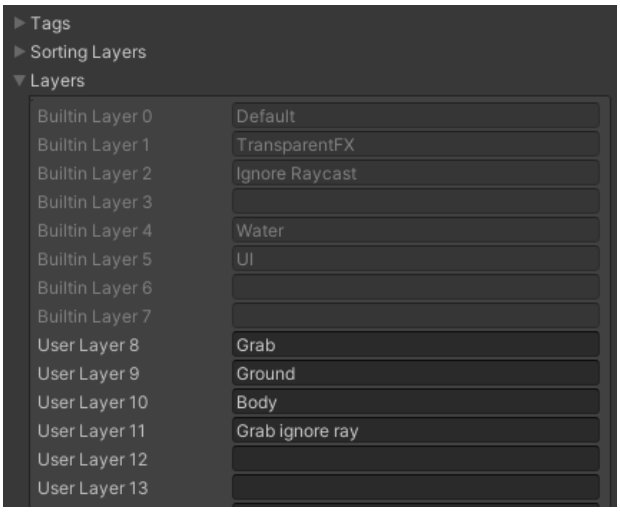
For every object that we want to be grabbable we need:

- **Rigid body**
- **Collider**
- **Xrgrabinteractible script**
- Reference video: https://www.youtube.com/watch?v=FMu7hKUX3Oo
- Set the object in the grab layer or grab ignore ray layer in order to be grabable by both the hand and the ray interactor or only by the hand.

Step 2. Disable interaction between body and object- set into a different layer

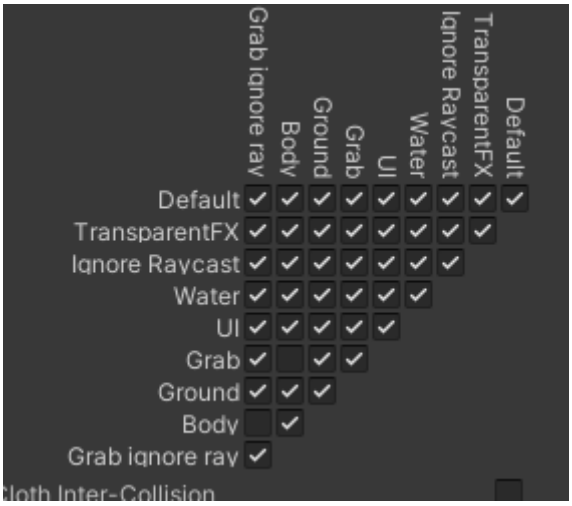
In case of a new project create the following layers

Layers:



Go to project settings and on graphics tab

- Set VR rig to body layer
- Set object to grab or grab ignore ray layer
- Go to edit project settings physics and in layer collision matrix uncheck appropriate interactions between the two layers.
- Set movement type to kinematic or velocity tracking



Step 3. To create a different anchor- snapping point to grab an object from.

- Import the object into the scene
- Add a **rigid body**
- Add an **XRGrabinteractible**
- Add a **box collider** onto the object and scale it to an appropriate size
    - If you put the collider on a child of the object where you have the xrgrabinteractible, you can drag and drop all colliders in the collider list in the xrgrab interactable component.
- Set the object to the grab/ grab ignore ray layer.
- Create a new **GameEmpty** object called pivot and set it as a child
    - Make sure the coordinate system is set to local and pivot
    - For the pivot the blue axis represents the forward axis and the green axis represents the up axis
    - If the object has a handle, center the pivot at the center of the handle and rotate appropriety.
    - Drag the new pivot gameboject into the attach transform reference inside the xr grab interactable

Note1.Grab object from a distance with ray interactor

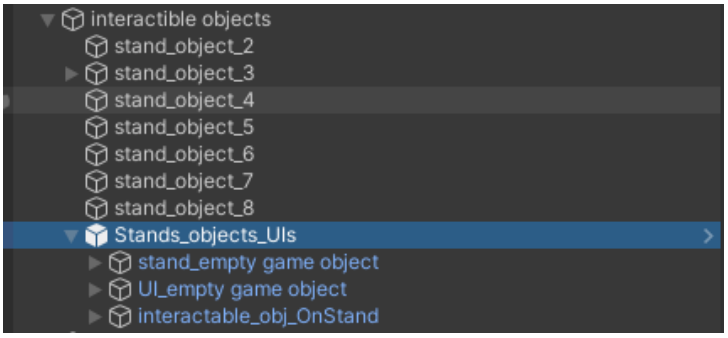
Select desired object and set it to layer grab

Note2.Grab Object only with direct interaction with hands

Select Desired object and set it to layer grab ignore ray

5. Interactible objects.

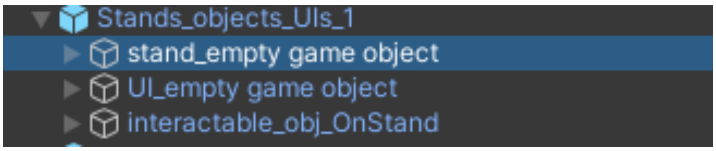
In this section of the hierarchy they are organized all interactible objects that have stand functionalities.

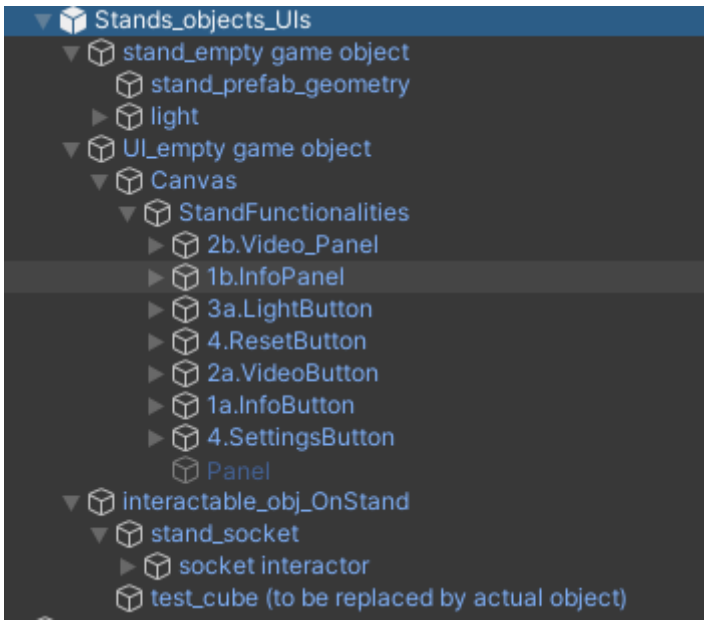


5aStands_objects_Uis

The stand_object_Ui consist of 3 main groups:

- **Stand_empty game Object**- is all related geometry with the stand for exhibition purposes.
- **UI_empty game object**- is all related UI elements and interactions that relate to user input, panels, video projections and functionalities.
- **Interactable_obj_OnStand** are all related functionalities needed for an object to be placed and work properly upon a stand, such as a:
    - Snap location so it hovers in mid air
    - A respawn location so the player can press the button and respawn the object at the starting location.





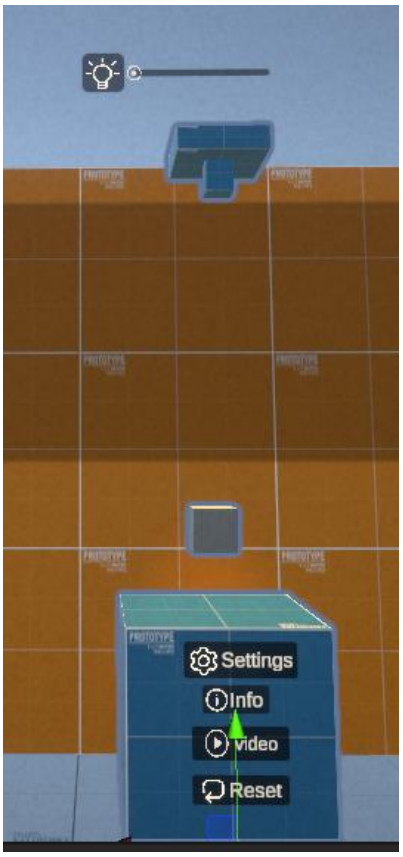
The stand objects Ui consist of the following elements:

5ai.Stand_prefab_geometry

Stand_prefab_geometry, is the cube stand that appears in the scene

5aii.Light Child Object

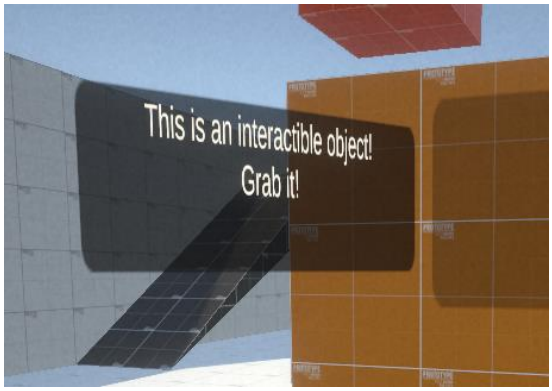
Light child object, is the light geometry along with the light above the stand that appears in the scene

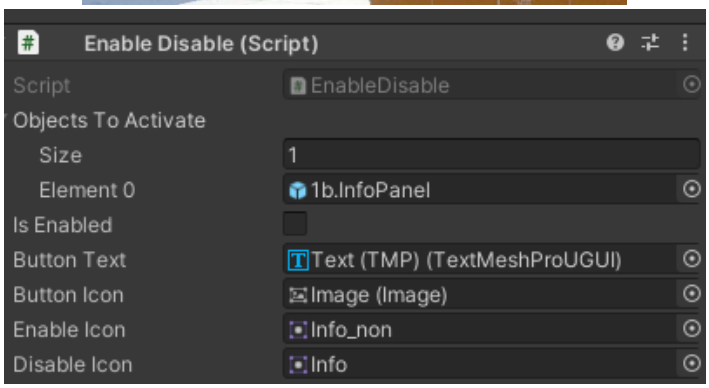


5b.UI_empty game Object

This GameObject has all functionalities to control **an info panel**, **a video panel**, **a reset button** to reset the translation on the object that appears on the stand and finally the **light on/off button** along with the slider that controls the intensity.

1a. InfoButton



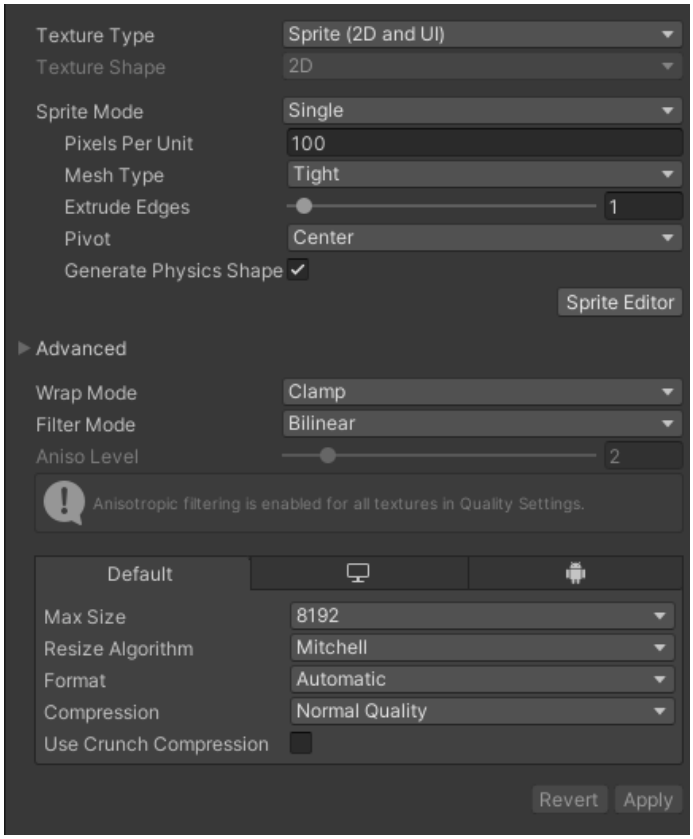


The **infobutton** opens up the **infoPanel** (**1bInfo.Panel**) gameobject.

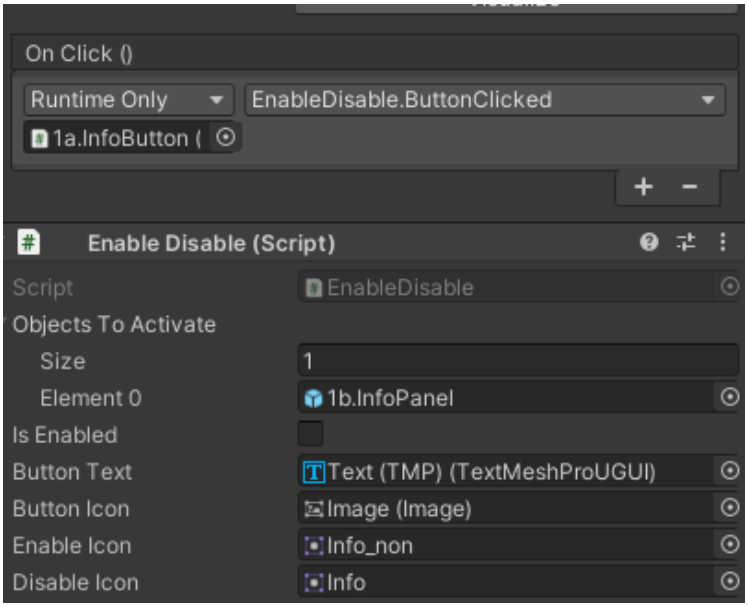
The infobutton has attached onto it a script called **enable disable** with the following references

- **Objects to activate**- a list of all elements that we would like to activate on/off in case we have more than one infopanels that work with the same button.
- **Button Text** a reference to the text object of the button

- **Button Icon**, a reference to the image object of the button
- **Enable icon**, a reference to the sprite icon that is used as enable
- **Disable icon**, a reference to the sprite icon that is used as disable
    - All sprites in the scene are from the website a noun project. In order to use standard png images as sprites, select the imported into the assets png image and on texture type select sprite (2D and UI)



- Finally on the on click event, there is a reference to the infobutton and the enabledisable.buttonclicked method from the enable disable script.



1b.InfoPanel

The infoPanel is a panel with a textMeshPro object with information concerning the object. Apart from the text object, the user can add also an image object to reference an image.

2a. Video Button

The video button incorporates the same functionalities with the infoButton with the only difference it enables disables the **2b.videoPanel** GameObject. Also in the enable/disable icons it has a different set of sprites.

2b. Panel

The Video Panel gameobject is a gameobject capable or reproducing a video sequence.

More info here: https://www.youtube.com/watch?v=OdVGyV3rTGs&t=318s

In orde to do that child to it there are two gameobjects a RawImage and an Audio Source

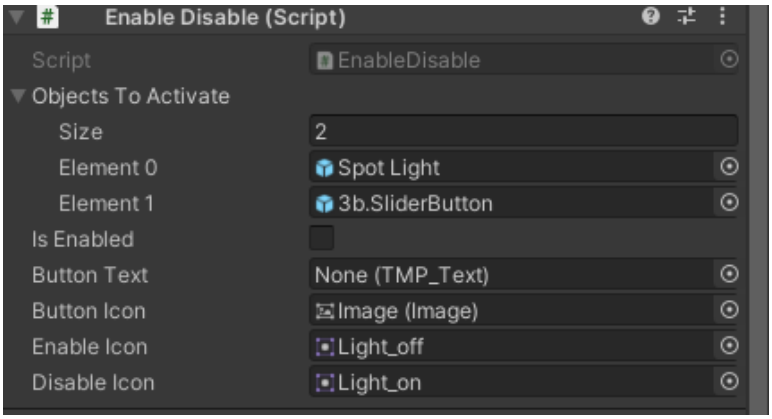
- The audio source on the spatial blend settings you need to change the slider from 2d to 3d so that it works in 3d space.
- To incorporate a custom video you need to follow the following steps
    - Step1. Import the video into Unity assets.- select the video and check on the inspector the videos resolution.
    - Step2. Create a new CustomRender Texture by right clicking to the assets window and set the resolution (size) to the same resolution as the video that you imported
    - Step3 on the video panel game object you will see a rawImage and an audioSource. Make the raw image the size of the canvas which is the size of your resolution (ex. 1920 by 1080) and then scale down holding shift in order to keep the proportions.

    - Step4. Select the Video_panel and add a video player Component on the inspector, on the render texture reference the video custom render texture we created above. Also add the video that you want to be playng in the Video Clip slot. Finally reference the audio source game object in the audio source slot in the video player component.
    - Step5. On the Raw Image gameobject reference the video custom render texture onto “texture” in the raw image component.

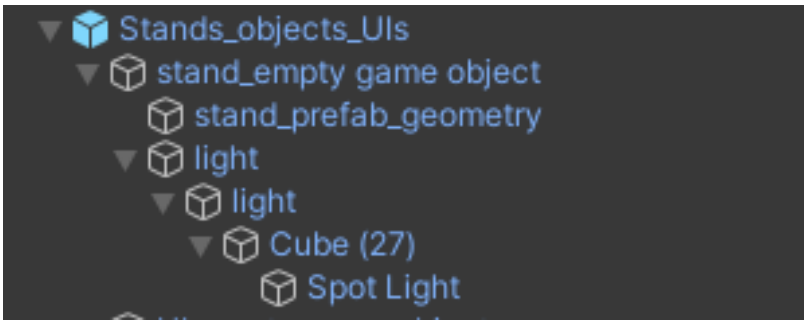
3a. Light button

The light button functionality includes an on/off button and a slider that controls the intensity of the light in question.

The light button includes as all buttons the enable disable script with reference to the spot light and **3b. sliderbutton** and the on off sprites.



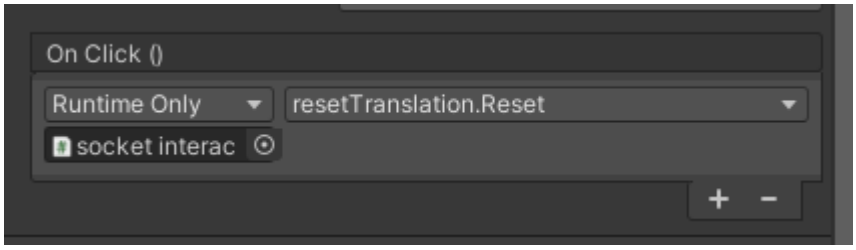
The spot light game object is located under the **stand_empty game** object as a child to the light prefab.



Finally the on click event we reference the light button and the method enabledisable.buttonClicked.

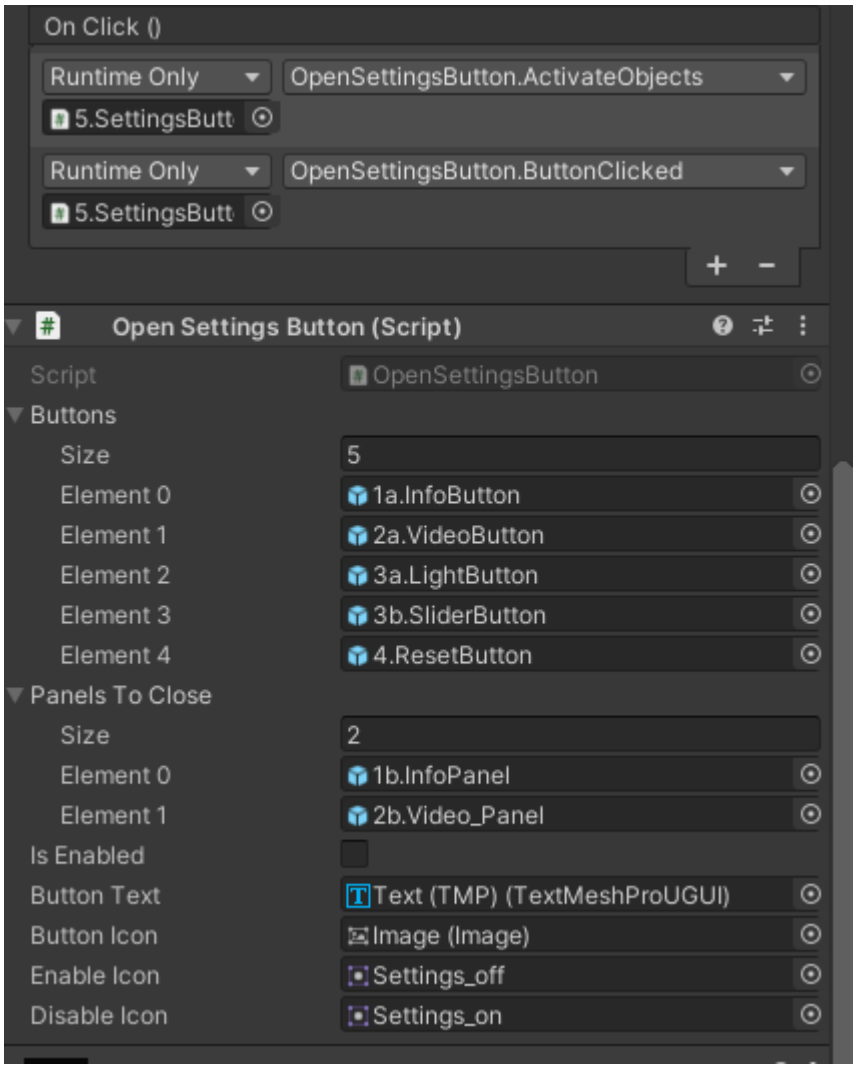
4.ResetButton

The reset button is used to reset the translation of the referenced object that appears on the stand in case it is lost. As it is a simple onclick button we simply reference the socket interactor gameobject that we will explain bellow



5.Settings button

The settings button is a simple on/off button that expands all functionalities explained above, for the player to be able to access them. Again, here we have referenced the open settings button in order to get all components and gameobjects that we want to turn on/off.



The **OpenSettings** button script has **2 main lists**:

- one buttons list that controls which objects are opening or closing and
- one panels to close list that controls which panels are opening and closing.

Both lists are parametric so they can be expanded or minimized.

Finally, on the **OnClick ()** event we reference the setting button in order to activate the **activate objects method** and the **buttonclicked method** inside the open settings button script.

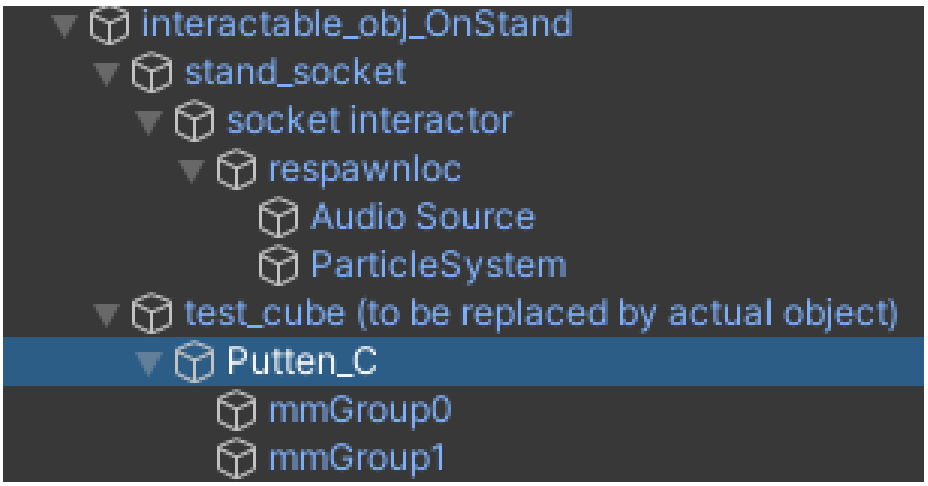
!!Important note!!

The stands_objects_UI and all blue colored objects are big prefabricated elements that incorporate all functionalities. In case the user needs to adapt- add or remove panels and functionalities:

- right click on the prefab on the hierarchy
- unpack prefab
- start copying or removing functionalities, adding the necessary references where needed.

5c.Interactable_obj_onStand

This gameobject holds all functionalities that allow for any object placed on the scene to hover above the stands and allow the user to interact with.



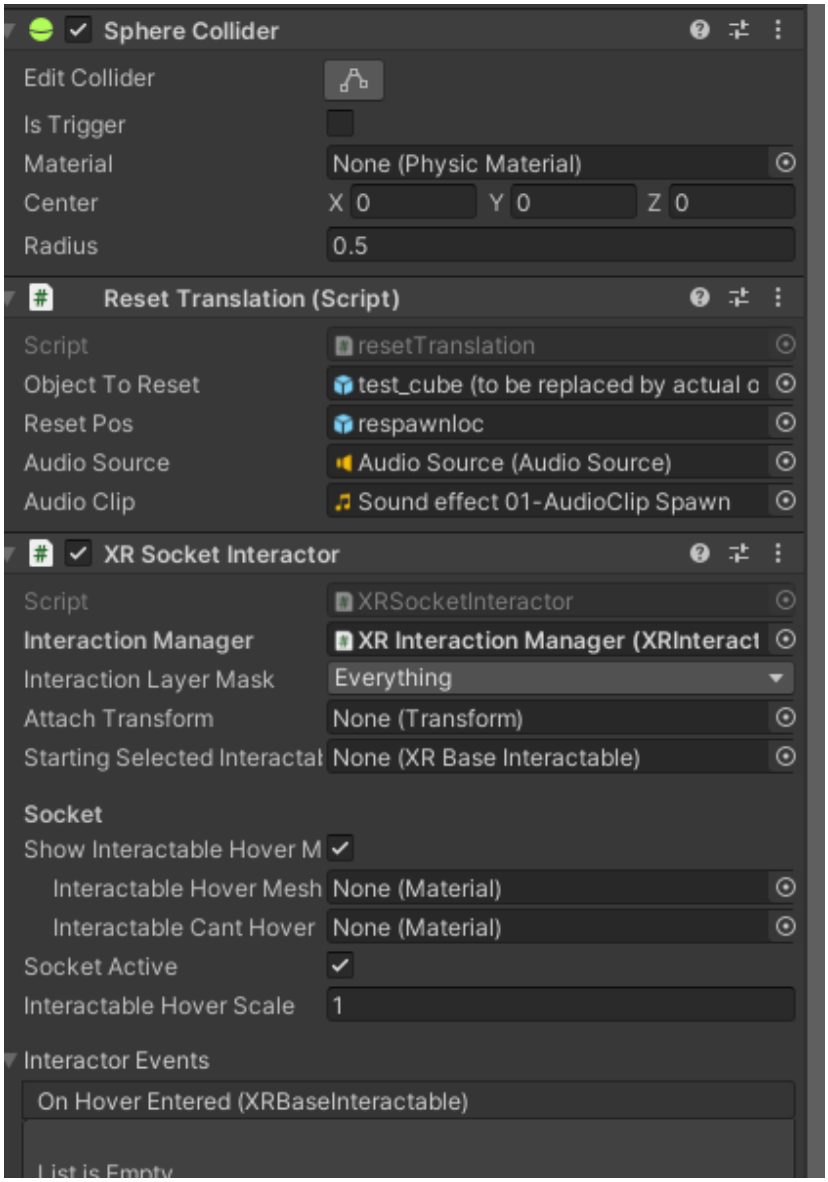
5ci.Socket Interactor gameobject

The first game object that holds specific functionalities is the socket interactor, that acts as a snap point. More specifically it holds the reset translation script that is connected to the reset button as seen above. The reset button resets the translation of the attached gameobject that we want to return onto the stand, if the player has lost it in the scene. In order to work properly we need:

- a sphere collider
- the reset translation script, which requires the following references
    - object to reset- a reference to the object we want to reset to original position
    - reset position- a reference to an empty game object that corresponds to the original position
    - audio source- a reference to the audio source gameobject in the hierarchy

- audio clip- a reference to the sound effect we want played when we initiate the Reset Translation Script

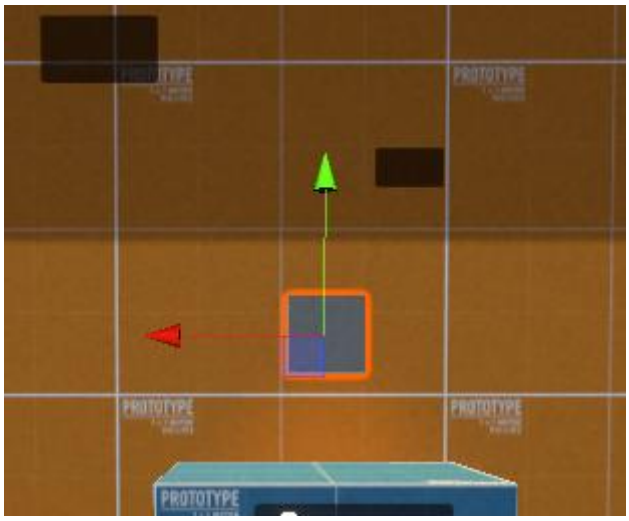
- the xr socket interactor script- the component responsible for the snapping behaviour



5cii.RespawnLoc

The respawnlocation is a simple game object that marks the translation of the location of where the respawn will happen. It holds an audio source and a particle system for effect.

5ciii.Test_cube (to be replaced by actual object)



The test_cube gameobject is an example object that needs to be replaced by the actual gameobject that will be placed on the scene.

In order to function as an interactable object, it needs the following components:

- Box collider
- Rigidbody
- Xrgrab interactable

!Important information about multi mesh vs single mesh objects!

If the gameobject consist of a single mesh there wont be any problem in working with the wrist inventory. But in the case that the gameobject consist of a multi part model please follow instructions in the **"Important notes about the WristSocket-single vs multi mesh object functionality"** on section 3c of this guide.

In brief, you can either replace the gameobject withyourown, or regardless of mesh count, set your gameobject that you want to interact with, as a chield to the cube, and turn off the mesh renderer component. Do not forget to resize the cube and the collider of the cube in order to fit the new object into its volume. By that way the wrist socket inventory system will work as intended.

6.PlayerObject category

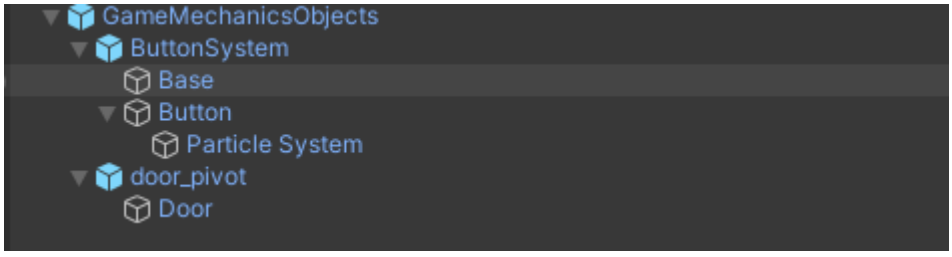
Here are all game objects that are part of the environment and do not have necessarily a prefixed stand functionality. This doesn't mean that stand functionalities such as info panels cannot be transferred into it.

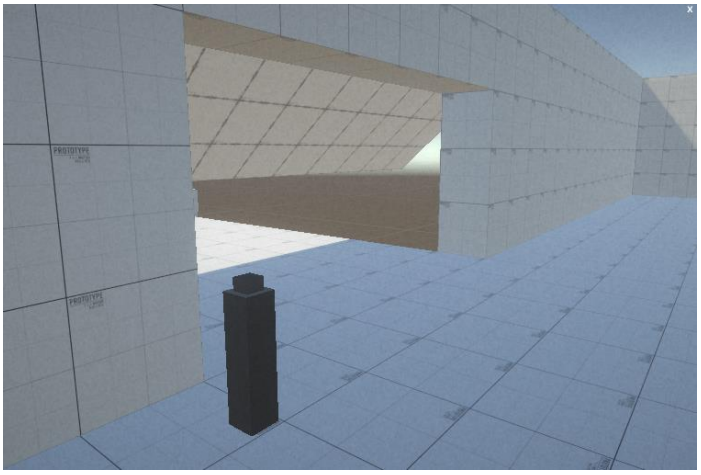
The main object in the hierarchy is a mesh model of a Traditional Cypriot House in the neighborhood of Strovolos, which was created by photogrammetry. The most important aspect that the user needs to keep in mind, is that if he will incorporate into the scene elements that come from photogrammetry they need to be up to or below 1mil polygons.

Also, if we want to incorporate the functionality of walking/teleporting inside, such as, the interior of a house or a town square, the ground mesh needs to be a separate mesh object as it needs the teleport area with fade script attached onto it, as seen above.

7.GameMechanics Objects

The gamemechanics object include specific mechanics that can be incorporated into the scene and use them appropriatly such as a door and a button system



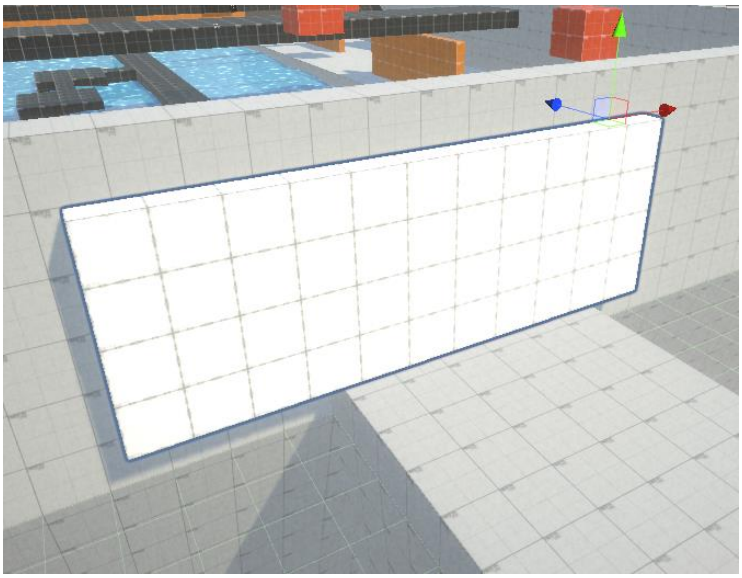


7a.Door_pivot

The door pivot game object is a simple geometry with an animation clip in order to simulate the way a door opens with the push of a button.

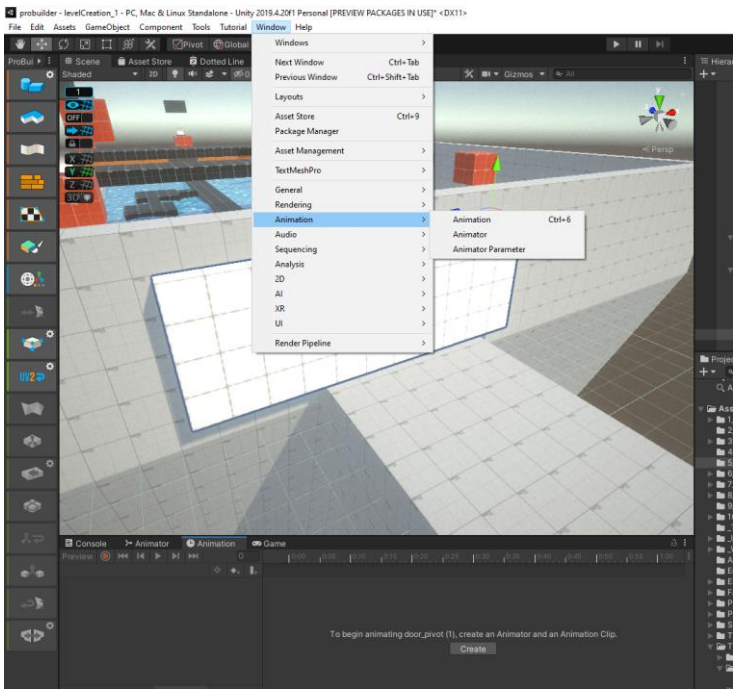
7ai.Create an open/close animation:

- Step 1. Select the object that you want to animate
- Step2. Create an empty gameobject and place it in the rotation axis position.

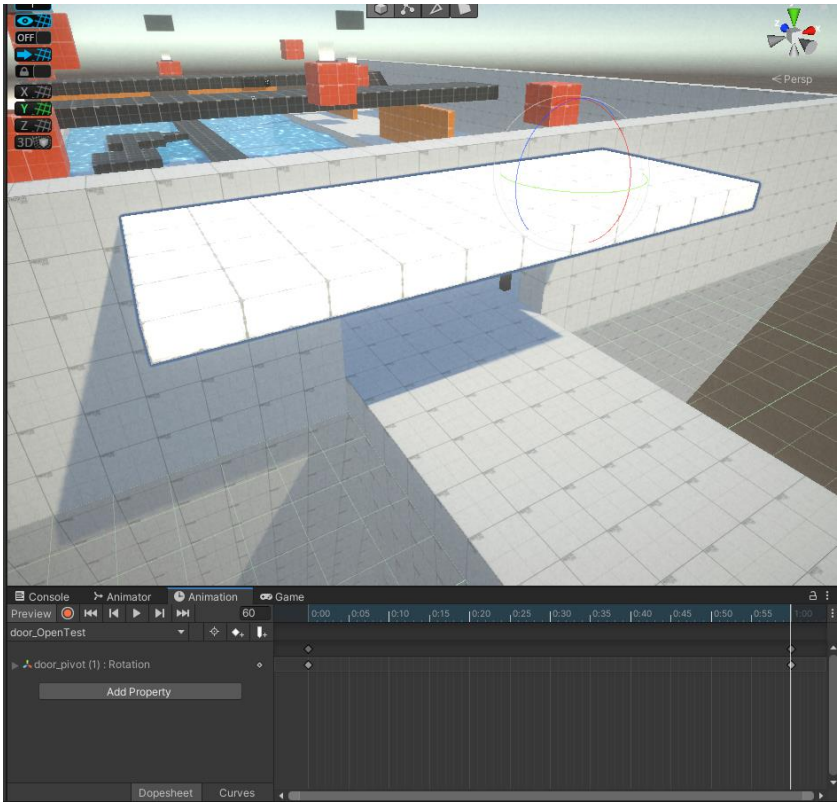


- Step3.place the door object as the child of the doorPivot

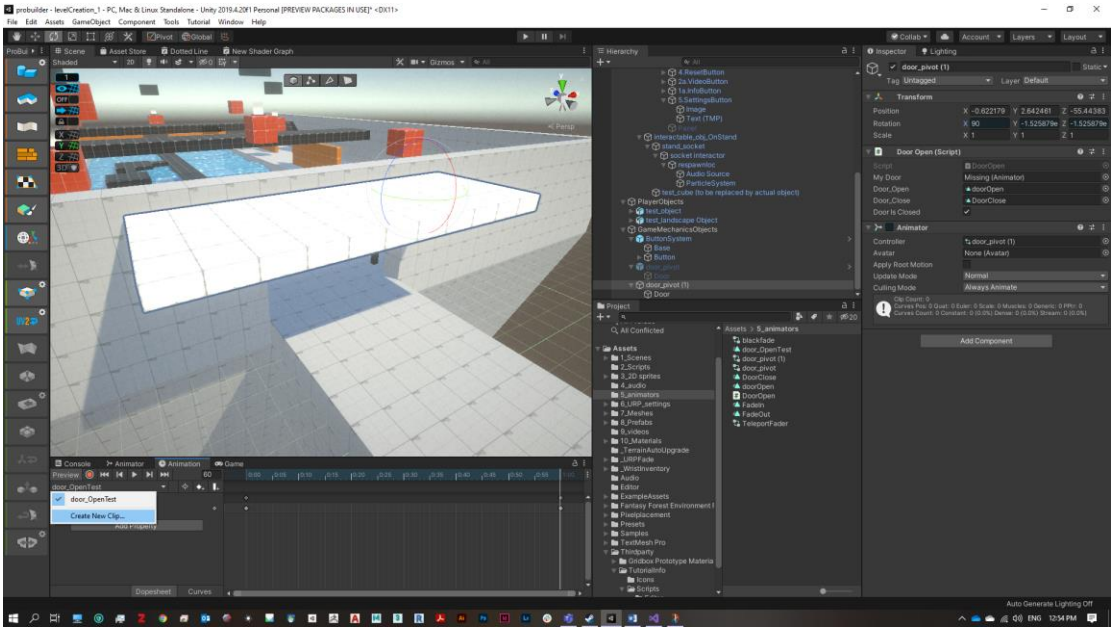
- Step4. Open the animator and animation panels



- Step5. Select the door pivot gameobject and on the animation Panel press create
- Step6. Press recond
- Step6. With record pressed go to frame 1 and rotate the door 90 degrees and turn off record

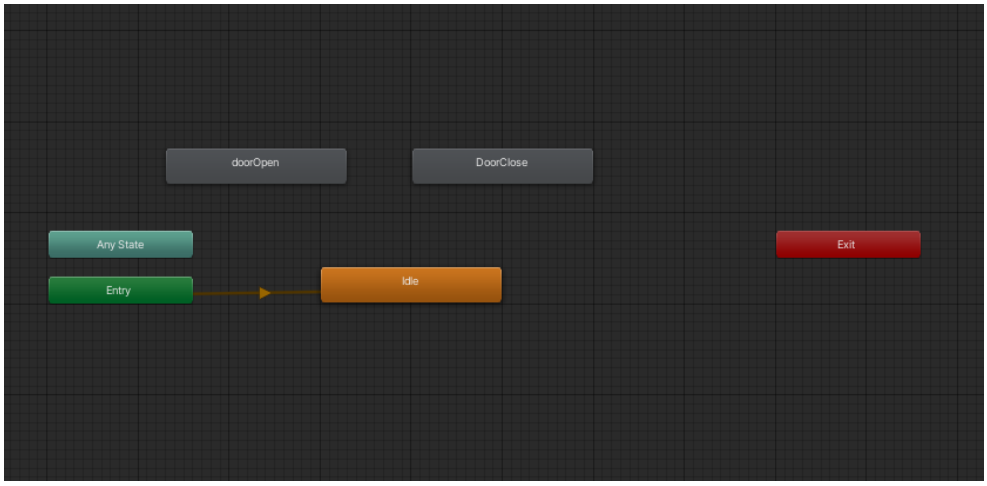


- Step7. Go to the drop down arrow in the animation clip name and press create new clip and name it Door_close

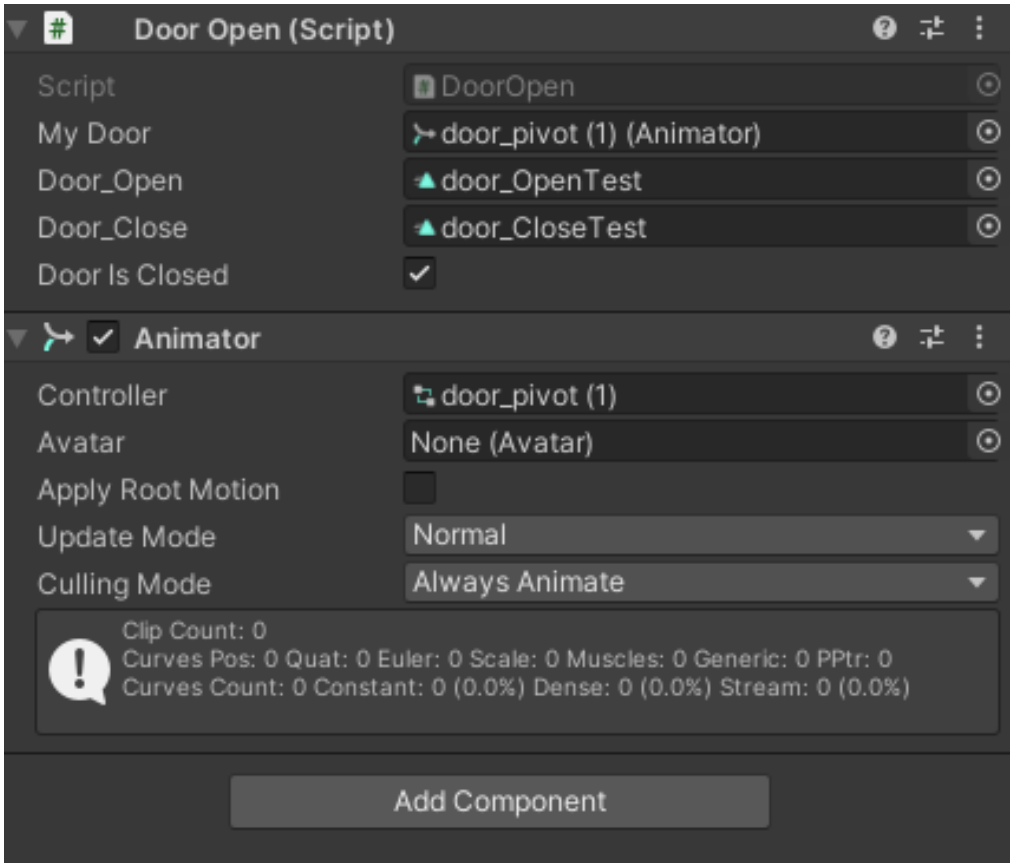


Do the same as before but in the opposite turn, start at frame 0 with the door opened and finish at frame 1 with the door closed.

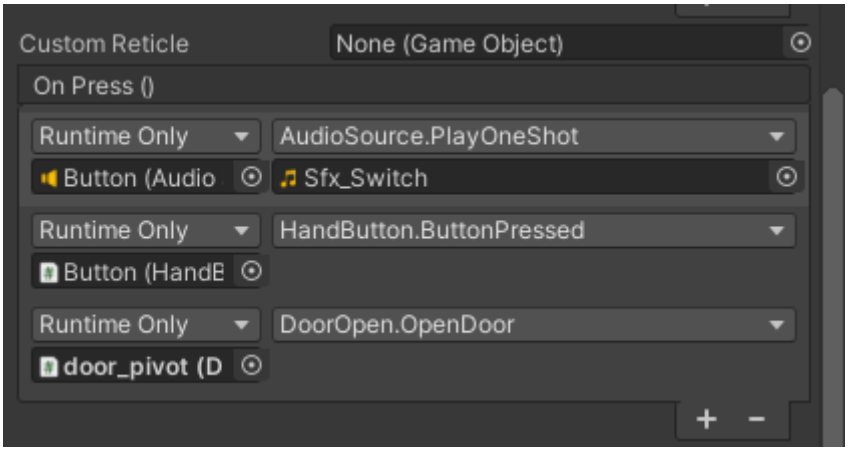
- Step8. Select the door pivot gameobject and open the animator panel. Right click and create new empty state, name it idle on the inspector and make it default by setting it layer default state (right click)



- Step9. Attach the script Door Open and reference all objects appropriately



- Step10. Select the button gameobject and on the handButton Script add the door_pivot reference with a reference to the method of doorOpen.OpenDoor

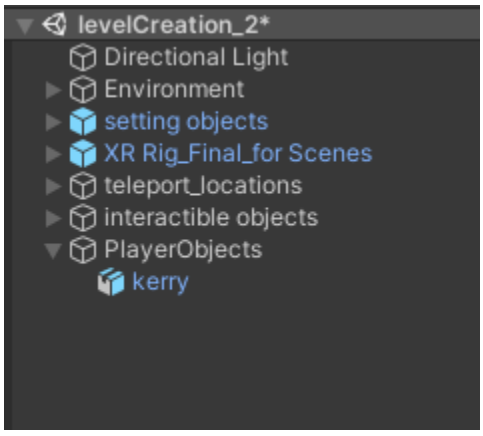


With this, every time someone presses the button it will activate the script to open close the door.

c.Scene: LevelCreation_2

The Level Creation_2 is a sample scene with a photogrammetry asset downloaded from SketchFab. It is a photogrammetry model of the Ballycarbery castle ruin[4]. The scene is organized as a virtual tour of the ruin. On the corner, there is a part of the previous scene, where there is a stand and a minimap with a video overview of the castle. The castle is teleportable and there are 6 teleport locations in various vantage points for better views.

i.SceneHierarchy

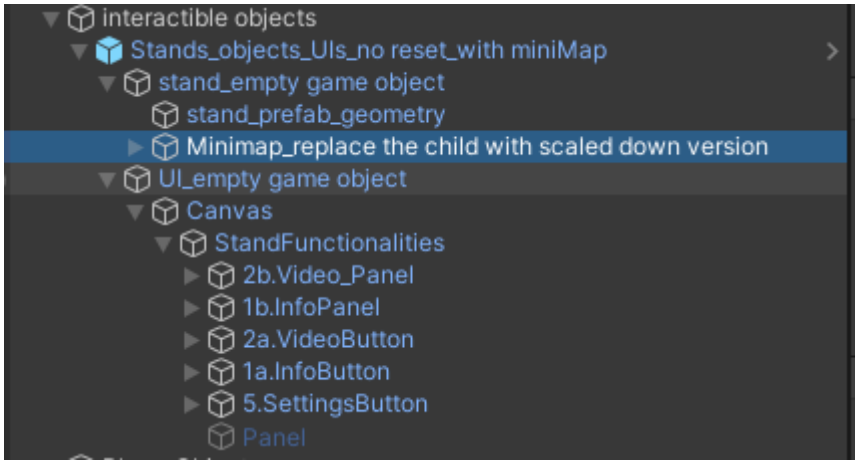


[4] https://sketchfab.com/3d-models/ballycarbery-castle-ruin-26ea562f32d54afe919b73486dbf7d53

ii.Scene Main Prefabs:

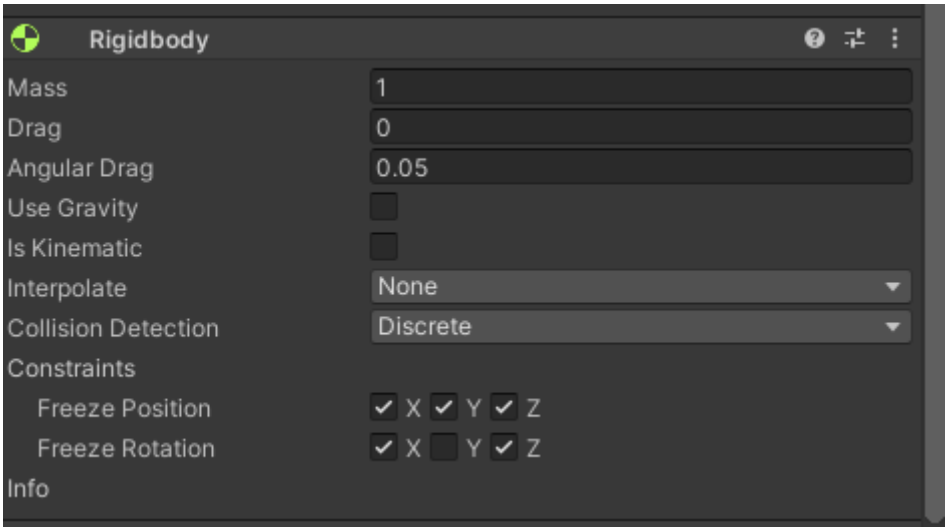
1.Interactible objects

The main differences from the previous scenes, are in the interactable objects. More specifically:



Instead of the typical stand functionalities as presented in the previous scene, in this scene it has been adapted to incorporate a mini map functionality in order for the player to be able to view and rotate prior to teleporting to specific locations.

The key difference is that apart from the xr grab interactable or simple interactable component attached to the castle gameobject, in the collider component we have the following constrains



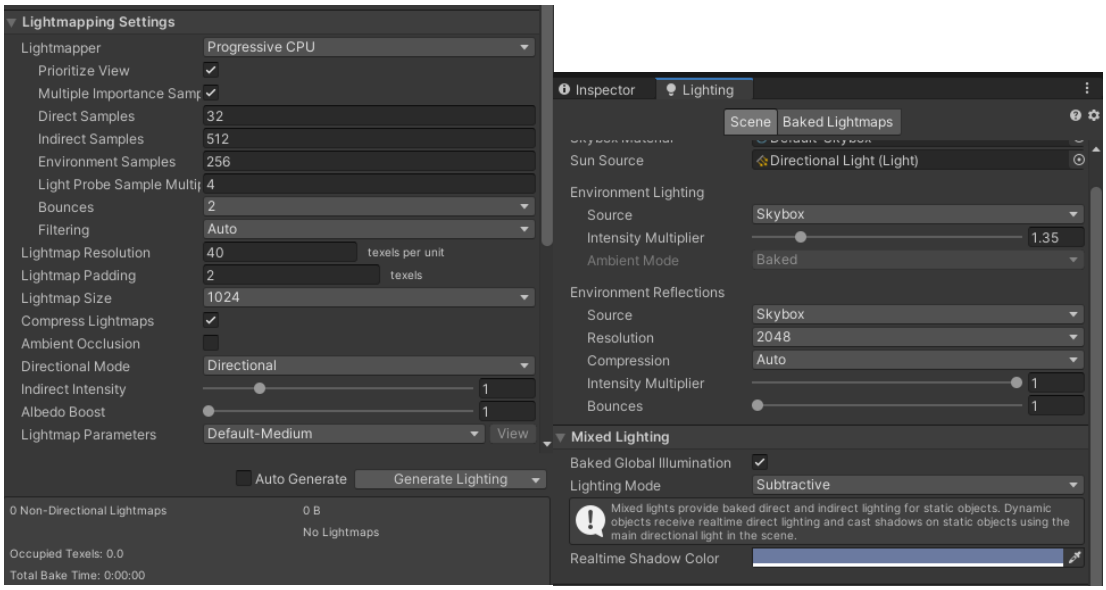
In this way we are able to interact with it but only rotate it in his axis.

The rest of the gameobjects in the scene are reconstituted elements adapted to the needs of this scene.

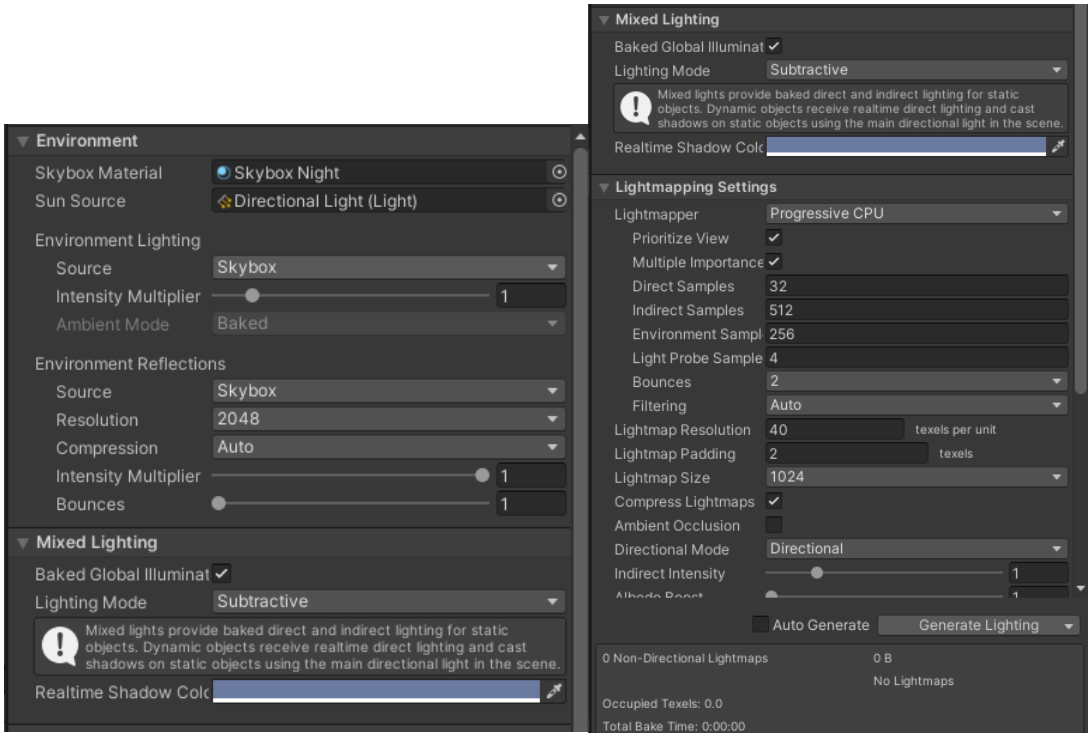
3.Final touches and creation of exportable exe file for use standalone

a.Lightmapping

Before creating the final scenes, all used scenes must be lightmapped. This means that all lights need to be baked

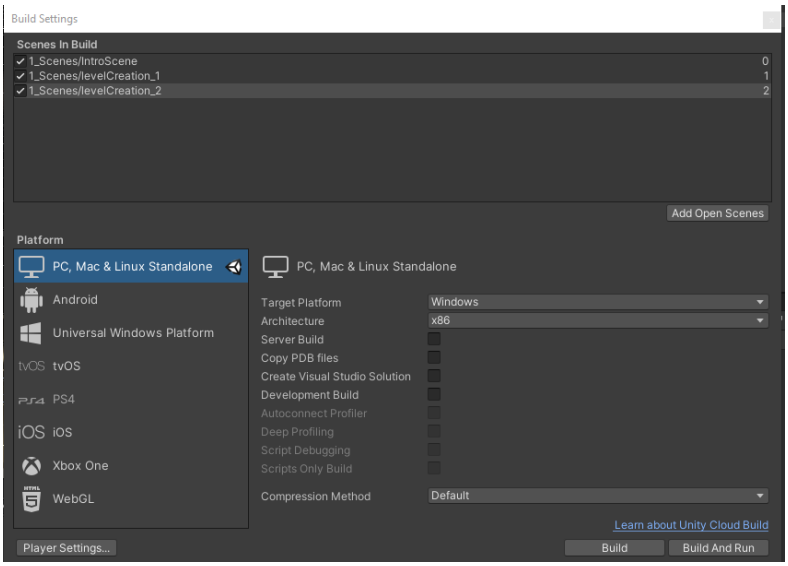


- Check in lighting panel
- Set Baked global illimitation to on and on debug settings press generate lighting.
- This will create lightmap data that are used in the baked lightmaps panel under the lighting tab, **do this for all active scenes.**



b.Building the executable file

- Go to project settings



- Check that all scenes that we want to incorporate into the executable, are added and checked.
- Check that the intro Scene is the first so that it loads first.
- Select the appropriate platform
- In case of a lot of postprocessing effects (under settings gameobject) you should avoid exporting for android. This also applies in case of a lot of mesh models with high polygon count.
- Select appropriate platform windows/ android etc
    - If android device, ex. Oculus quest 2 without Oculus link check under player settings the android api level to be 23 or higher.
- Press built/ built and run

C. Adding the downloaded file and recreating the project files in you local machine

- Download all zip files
- Unzip the first zip file named ovret_1, the unzipping process shoud continue thereafter.
- Download unity hub
- Add the appropriate version of unity via the unity hub or if it is not available from the website of unity
    - Include also android packages during install along with visual studio files
- Go to the main screen where all the project files are located and press add, and reference the location of the unzipped project folder
- Open unity